Wolfgang Keller

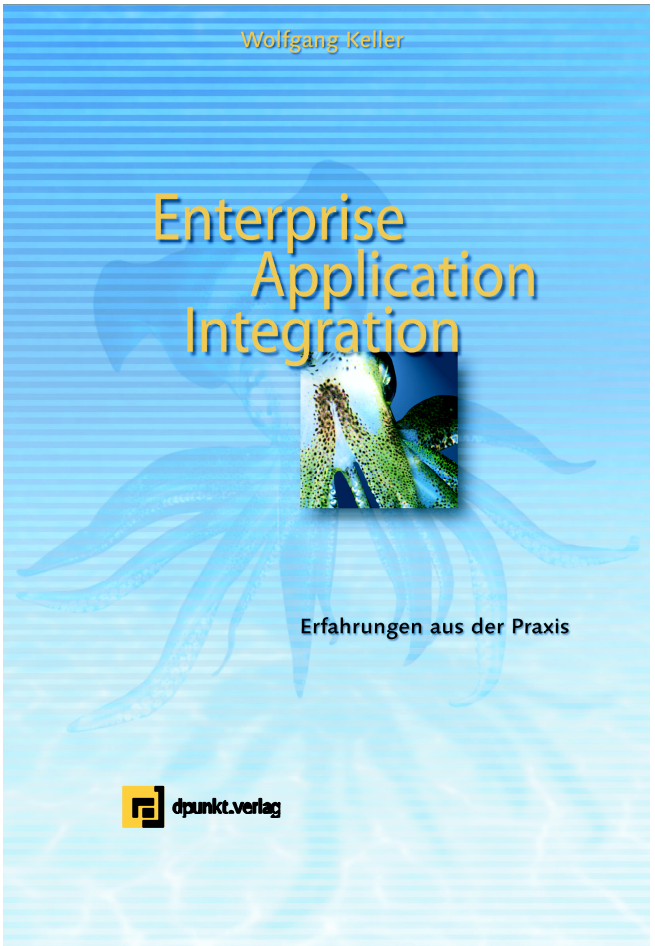
Enterprise Application Integration

Erfahrungen aus der Praxis

dpunkt.verlag
ISBN 3-89864-186-4

Leseprobe / Auszug

Hinweis: Es handelt sich um Auszüge aus der letzten aktuellen Arbeitsversion. Die Qualität entspricht noch nicht der Druckqualität. Voraussichtliche Auslieferung ab ca. 25.6.2002, Zu beziehen zum Beispiel bei amazon.de unter
http://www.amazon.de/exec/obidos/ASIN/3898641864/objectarchite-21/



Vorwort

Für dieses Buch über Enterprise Application Integration (EAI) hat es zahlreiche Anstöße gegeben. In unserer Firma, der Generali Gruppe Wien, ist wie bei vielen anderen Finanzdienstleistern auch, eine selbst gebaute EAI-Lösung entstanden. Etwas später wurde in anderen Teilen des Konzerns für ein zentrales E-Business-Projekt ein EAI-Produkt ausgewählt und später als Konzernstandard etabliert. Seit dem Jahr 2000 wird der Begriff EAI im Unternehmen immer wieder gebraucht und auch missbraucht. Um das Bild der Verwirrung zu komplettieren, haben wir seit 1995 auch Workflow-Lösungen im Haus. Diese treten allerdings immer mehr in Konkurrenz zu den Fähigkeiten von EAI-Integrationsservern. Die Unterschiede sind nicht trivial. Dazu kommen bei einem typischen Finanzdienstleister noch Ad-hoc-Workflows auf der Basis von Groupwarelösungen wie Lotus Notes. Doch damit nicht genug. Die meisten Finanzdienstleister setzen massiv auf IBM Host-Produkte wie OS/390, CICS und IBM MQSeries, während das Projekt, das den Konzernstandard gesetzt hat, mit solchen Umgebungen nur am Rande zu tun hatte.

Es wurde also Zeit, die Landkarte zu bereinigen und strategische Produktentscheidungen zu treffen. Wenn man das tut, möchte man sich typischerweise auf Referenzmodelle oder sonstige Unterlagen und Publikationen verlassen, die einem eine Richtlinie geben, wie man bei der Bereinigung sinnvoll vorgehen könnte. Bei der Literaturrecherche dazu war schnell festzustellen, dass hier eine Marktlücke existiert. Es gibt nur wenige Bücher zum Thema EAI (zum Beispiel [Lint2001, Ruh+2001]) und auch nur einen deutschen Titel [Nuss2000]. Während dieses Buch geschrieben wurde, sind noch zwei Spezialtitel zum Thema EAI mit J2EE auf den Markt gekommen [Juric2002, Shar+2001]. Alle diese Publikationen haben die Frage nach einem sinnvollen Referenzmodell nicht vollständig beantwortet und als Finanzdienstleister findet man sich auch nicht immer mit seinen Problemen wieder.

Damit war die Idee geboren, diese Lücke zu schließen und dieses Buch zu schreiben.

Danksagungen

Als derzeit nicht kodierender Manager war ich darauf angewiesen, auch die nötigen Mengen an Basisstoff zu bekommen, um dieses Buch schreiben zu können. Ich möchte dafür den Herren Rainer Frömmel (Shark Soft Wien) und Bernhard Anzeletti (Generali Wien) danken, die EDS gebaut und mit der Euro-Umstellung im großen Maßstab in Produktion gebracht haben. Wichtige Impulse habe ich außerdem von allen erhalten, die von „Schnittstellen zum Nulltarif“ geträumt und mit dieser Vision für kreative Verwirrung gesorgt haben, die ich unter anderem mit diesem Buch wieder einfangen muss. Mein Vorgesetzter, Herr Walter Steidl, hat freundlicherweise die Freigabe für das Buch gegeben. Mein besonderer Dank gilt allen Reviewern, besonders Manfred Polczer, Dr. Gernot Starke, Markus Völter, Dr. Gabriele Wellm und dem Verlagsteam, vertreten durch Frau Christa Preisendanz.

Wien im Frühjahr 2002

Wolfgang Keller

Über den Autor

Wolfgang Keller ist Manager für die wiederverwendbaren Softwarekomponenten (Plattformmanager) der Generali Group Vienna. Sein Verantwortungsbereich umfasst technische Basissysteme für die Phoenix-Produktlinie, Produktarchitektur und internationale Koordination der Entwicklungslabors für die Generali-Plattform, sowohl für fachliche als auch für technische Anwendungen, die das Kerngeschäft von Versicherungen unterstützen.

Er studierte nach einer "Siemens Stammhauslehre" Informatik/BWL an der Technischen Universität München und war vor seiner Tätigkeit bei der Generali als Seniorberater und Projektmanager bei der software design & management AG in Hamburg und München beschäftigt. Herr Keller hat über lange Zeit die VAA-Initiative des GDV (http://www.gdv.de/vaa) beraten.

Website: www.objectarchitects.de
E-Mail: wk@objectarchitects.de

Inhaltsverzeichnis

1 Einleitung und Überblick

Enterprise Application Integration (EAI) und Enterprise Nervous Systems (ENS) sind Technologien, die zum Baukasten eines Softwarearchitekten auf Unternehmensebene gehören. Neben erheblichen Kostensenkungs- und Beschleunigungspotenzialen durch den Einsatz dieser Technologien gibt es aber, wie bei jeder Technologie auch, Risiken, die durch einen falschen oder nicht angepassten Einsatz solcher Werkzeuge entstehen.

Viele Unternehmen können mit EAI Geld verdienen

Dieses Buch entstand aus der Beschäftigung eines "Unternehmensarchitekten" aus der Versicherungswirtschaft mit dem Thema EAI und ENS. Ich habe es vor allem geschrieben, weil ich ein solches Buch gebraucht hätte, um in der Lage zu sein, die Angebote verschiedener Hersteller zu beurteilen und die Frage "make or buy" profund zu beantworten.

Am meisten von diesem Buch profitieren Softwarearchitekten von Finanzdienstleistern. Aber auch Softwarearchitekten aus anderen Branchen werden profitieren

Das Buch hat daher vor allem den Anspruch, Softwarearchitekten auf Unternehmensebene im Bereich der Finanzindustrie zu helfen. Wenn Sie als solcher arbeiten, werden Sie von der Lektüre am meisten profitieren. Die Beispiele sind bewusst auf das Arbeitsgebiet des Autors fokussiert, weil der praktische Wert eines Buches dann höher ist, wenn echte Projekterfahrung hinter einem Text steht und nicht vor allem theoretische Überlegungen.

Softwarearchitekten aus anderen Branchen werden jedoch ebenfalls profitieren. Die Zielgruppe sind also vor allem Entscheider, die über den Einsatz von EAI-Technologien mitbestimmen, und Unternehmensarchitekten, die dafür die Umsetzungspläne entwerfen.

Sie können diesen Text natürlich auch zur Einarbeitung verwenden, wenn Sie als Softwarearchitekt eines anderen Projektes mit EAI-Technologien zu tun haben oder wenn Sie ein EAI-Konzept beurteilen möchten.

Das Buch ist von Profis für Profis gemacht

Explizit angesprochen sind vor allem Leser, die mit Grundbegriffen der Datenkommunikation, wie Queuing-Modelle, TCP, SNA, XML und Ähnlichem, etwas anfangen können und Berufserfahrung als Softwareentwickler im Umfeld großer Geschäftsinformationssysteme haben. Viele Bücher zu EAI werden deshalb so extrem dick, weil sie genau alle diese Grundbegriffe erklären. Dieses Buch versucht dagegen, sich speziell auf die neuen Aspekte von EAI und ENS zu konzentrieren, die Sie als Entwickler und Architekt benötigen, um sich schnell einzuarbeiten, und ist daher bewusst dünner als viele andere Bücher in demselben Gebiet, die zusätzlich noch Grundlagen der Datenkommunikation mit vermitteln.

Dadurch ergibt sich auch ein gewisses Dilemma: Experten sollten diesen Text schnell diagonal lesen können und nur diejenigen Abschnitte durcharbeiten, die Aspekte abdecken, die sie nicht sowieso schon kennen – daher wird nicht jeder Begriff wie zum Beispiel

"Application Server" oder XML von null ausgehend eingeführt. Stattdessen wird er erst einmal verwendet und es werden dann Querverweise auf Erläuterungen und Definitionen an geeigneten Stellen angebracht. Das mag zwar methodisch nicht immer 100% sauber sein, erleichtert aber das schnelle Lesen für die Zielgruppe.

1.1 Was wissen Sie, wenn Sie dieses Buch gelesen haben?

Wenn Sie dieses Buch gelesen haben, kennen Sie ***wichtige Anwendungsfelder***, bei denen es sich potenziell lohnt, mit Methoden der Enterprise Application Integration zu arbeiten (siehe vor allem Kapitel 2). Sie kennen auch den Unterschied zwischen EAI und B2B-E-Commerce (Business to Business: Elektronischer Handel zwischen Unternehmungen, meist über das Internet), wissen aber, dass dafür sehr ähnliche Technologien eingesetzt werden, allerdings mit einem anderen Fokus.

Sie kennen die ***wesentlichen Entscheidungsfelder, in denen Sie Architekturentscheidungen*** treffen müssen, wenn Sie eine Enterprise-Application-Integration-Lösung für Ihr Unternehmen entwerfen wollen (siehe Kapitel 3 und 4).

Sie kennen wesentliche Fähigkeiten von Integrationsservern (siehe wieder Kapitel 3, 4 und auch Kapitel 10 als kondensierte Checkliste) und können damit beurteilen, ob Sie solche Fähigkeiten mit in Ihre Evaluierung von EAI-Produkten aufnehmen wollen.

Sie haben in Kapitel 5 ***zwei Fallstudien*** gesehen, die Ihnen einen ***Vergleich zwischen einer selbst gebauten EAI-Lösung*** *(Abschnitt 5.1)* ***und einem kommerziellen High-End-Produkt*** (Abschnitt 5.2) erlauben. Sie lernen außerdem die EAI-Strategie von Microsoft kennen (Kapitel 6) und wissen, wie .NET und EAI zusammenspielen. Dabei erfahren Sie auch etwas über die Rolle von neuen Konzepten wie SOAP und Webservices für EAI. Sie bekommen damit ein Gefühl, warum selbst gebaute Lösungen auf dem Rückzug sein werden. Für Fans von J2EE wird auch diese Architektur in Hinblick auf EAI beleuchtet (Kapitel 7).

Sie kennen typische "Verkaufsgeschichten" von EAI- und ENS-Verkäufern und können beurteilen, welchen Nutzen Sie wirklich in Ihren Business Case schreiben können und wo Sie es mit eher unrealistischen Versprechungen und Erwartungen zu tun haben. Diesem Thema habe ich Teile von Kapitel 8 „Praktische Erfahrungen mit EAI“ gewidmet.

Anhand einer groben Checkliste können Sie Ihren ***Weg zu einem EAI-Architekturkonzept*** (Kapitel 9) erarbeiten und haben es leichter, eine EAI-Lösung für Ihr Unternehmen zu entwickeln.

1.2 Wie können Sie dieses Buch lesen?

Sie können dieses Buch am besten diagonal von vorne nach hinten lesen und dort einhaken, wo Sie Dinge als neu und nützlich empfinden.

Es gibt aber auch Bereiche zum Nachschlagen, wie die Produkt-Checkliste (Kapitel 10) oder die sehr detaillierte Abhandlung von Kommunikations- und Integrationstechnologien (Kapitel 4), die Sie am besten nur bei Bedarf lesen sollten und wie gesagt zum Nachschlagen nutzen können, wenn Sie Ihr konkretes EAI-Projekt durchführen.

2 Wann brauchen Sie eine EAI-Lösung?

Wie Sie noch sehen werden, ist der Begriff EAI nicht gerade präzise definiert. Wenn Sie allerdings vom Verkäufer eines EAI-Tools angesprochen werden, wird der Ihnen klar machen, dass Sie genau sein Tool benötigen. Oft wird er das tun, ohne wirklich nach Ihrem Anwendungsfall zu fragen – one tool fits all.

Dieses Kapitel gibt Ihnen zunächst einen ersten begrifflichen Überblick über EAI. Sie werden dann die Trends kennen lernen, die den Bedarf nach EAI-Lösungen wachsen lassen.

Dann erfahren Sie Näheres zu vier häufigen Anwendungsfällen für EAI-Lösungen. Es sind dies:

?? Multichannel-Architekturen,

?? Application to Application (A2A) Integration,

?? Geschäftsprozessintegration mit Workflow und

?? Anwendungen aus dem Bereich Trade Rooms.

Das Kapitel stellt die rein durch das Geschäft verursachten Anwendungsfälle für diese Technologien dar. Über die Technologien selbst lesen Sie zunächst noch wenig. Es gibt sicher noch mehr Anwendungsfälle. Die hier aufgeführten und beschriebenen treten jedoch sehr häufig auf.

2.1 Was ist EAI?

EAI heißt Enterprise Application Integration. EAI-Produkte sind Software, die es erlaubt, die Anwendungen eines Unternehmens zu integrieren. Der Begriff wurde geprägt durch das Bemühen, viele Anwendungen, die nicht für eine Zusammenarbeit entworfen wurden und auch nur Teilaufgaben von Geschäftsprozessen abdecken, dazu zu bringen, in einheitlichen Geschäftsprozessen zusammenzuspielen. Es geht also darum, heterogene Anwendungen eines Unternehmens so zu integrieren, dass sie sich möglichst so verhalten, als wären sie von Anfang an dafür entworfen worden, die aktuellen Geschäftsprozesse eines Unternehmens zu unterstützen. Abschnitt 2.3.3 beleuchtet dieses spezielle Benutzungsszenario von EAI später noch näher.

Da die Werkzeuge, die wir in diesem Buch vor allem erläutern, nämlich so genannte EAI-Integrationsserver, auch in anderen Bereichen Verwendung finden, macht es hier Sinn, EAI von so genanntem B2B-E-Commerce kurz abzugrenzen.

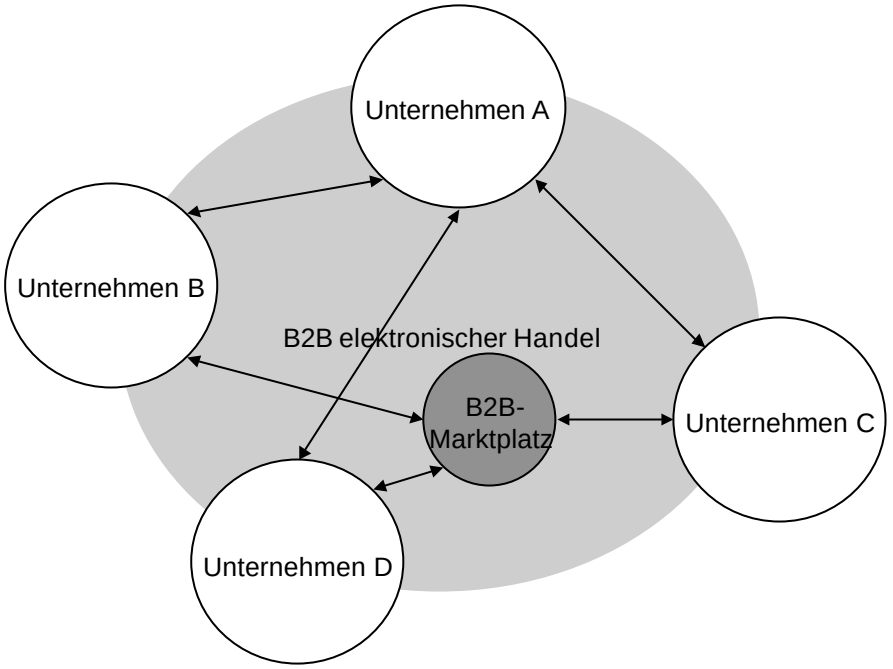


Abbildung 1: B2B-E-Commerce
Die Pfeile stellen den Informationsaustausch zwischen den Partnern dar.

B2B-E-Commerce rationalisiert den Handel zwischen Unternehmen. Mittels Geschäftsprozess-Reengineering gestalteten viele Firmen in den 90er Jahren ihre internen Prozesse neu. Dabei beschleunigten sie vor allem den Fluss von Gütern. Heute machen die Unternehmen den Fluss von Informationen zwischen Geschäftspartnern schneller. B2B-E-Commerce wird also vor allem eingesetzt, um schneller und mittels elektronischem Datenaustausch Geschäfte zwischen Unternehmen zu vereinbaren und abzuwickeln. Das ist nicht neu. Die Versuche gibt es mit EDI schon mehr als 20 Jahre. Nur durch neue Technologien, wie vor allem das Internet und die dort verwendeten Technologien wie XML, SOAP, Webservices und auch Integrationsserver, erhalten diese Bestrebungen neuen Schwung und die Unternehmen hoffen, bei diesem Anlauf mehr zu erreichen. Zum Teil wurde auch schon in bestimmten Industrien viel erreicht, wie die Geschichte von RosettaNet zeigt.

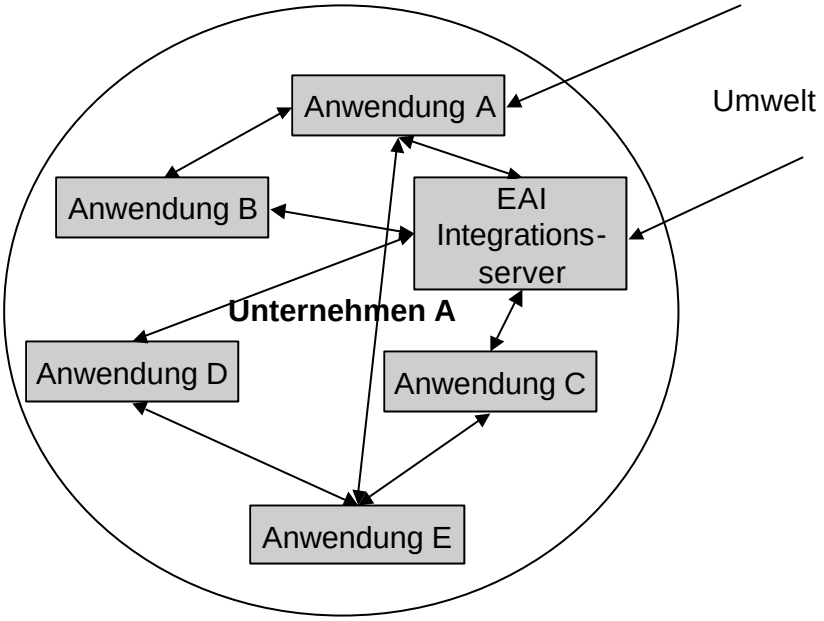


Abbildung 2: EAI
Die Pfeile stellen wieder den Informationsaustausch dar. Die Unternehmensumwelt kann entweder direkt mit einer Anwendung (A) kommunizieren oder die Kommunikation mit der Unternehmensumwelt wird über einen EAI-Integrationsserver konzentriert abgewickelt. Dieser kann auch als Hub & Spoke-Verteiler für Informationsaustausch im Unternehmen eingesetzt werden.

EAI (Enterprise Application Integration) beschleunigt und rationalisiert die Informationsflüsse innerhalb eines Unternehmens. Wenn man sich Abbildung 2 ansieht, wird man feststellen, dass sie strukturell identisch zu Abbildung 1 ist. Die Unternehmen wurden durch die einzelnen Anwendungen eines Unternehmens ersetzt, die miteinander kommunizieren müssen, und der B2B-Marktplatz durch den EAI-Integrationsserver. Damit zielt EAI auf die Rationalisierung innerhalb des Unternehmens.

Es ist einleuchtend, dass die Werkzeuge, die man für EAI und B2B-E-Commerce verwendet, strukturell ähnlich sind. Dieses Buch beschäftigt sich allerdings schwerpunktmäßig mit EAI. B2B-E-Commerce werden wir in einigen Abschnitten streifen, vor allem dann, wenn die Strategie von Microsoft erläutert wird. Deren EAI-Tool, der BizTalk-Server, stammt eher aus dem Bereich B2B-E-Commerce und es macht daher auch Sinn, dieses Gebiet im Überblick zu erläutern, wenn man ein gutes Verständnis für EAI erreichen will.

Die Werkzeuge, die für EAI entwickelt wurden, kann man für viele Anwendungsfälle in Unternehmen verwenden, so zum Beispiel um so genannte Multichannel-Architekturen aufzubauen (siehe dazu Abschnitt 2.2) oder um generell die Kommunikation zwischen Anwendungen eines Unternehmens einfacher und billiger zu gestalten (siehe A2A-Kommunikation, Abschnitt 2.3.1.3).

Das Akronym EAI ist zu einem Sammelbegriff für viele Kategorien von Softwarelösungen geworden, die sämtlich etwas mit der Integration heterogener Lösungen innerhalb eines Unternehmens zu tun haben. Trotzdem unterscheiden sich aber die optimalen Lösungen immer noch sehr stark, je nachdem, um welche Art von Problem es sich handelt – auch wenn sie alle unter den Oberbegriff EAI-Lösungen fallen.

Um abzugrenzen, mit welcher Art von Softwareprodukten und -lösungen wir uns hier befassen wollen, macht es Sinn, noch ein weiteres Akronym einzuführen. Das Akronym ENS

steht für Enterprise Nervous System. Gartner Research hat diesen Begriff eingeführt für die Summe aus:

- Integration Brokern – das sind Sternverteiler, die für die Kommunikation von Anwendungen genutzt werden, um über Adapter mit vielen Systemen kommunizieren zu können,
- Geschäftsprozessmanagementtools,
- Middleware zur Kommunikation,
- Webservern,
- Applikationsservern
- und Datenintegrations- und ETL[1]-Technologien.

Bei den oben genannten Kategorien von Software handelt es sich um technische Software ohne fachlichen Inhalt. Wenn man sich nun das Anwendungs- und Softwareportfolio einer typischen Versicherung ansieht (siehe auch Abbildung 3), dann findet man dort:

- Anwendungen, die das Kerngeschäft abdecken. Es sind dies meist große, hochspezialisierte Hostanwendungen, die man auf dem freien Softwaremarkt von ERP-Herstellern noch nicht zukaufen kann. Bei Versicherungen ist dies zum Beispiel ein Vertragssystem für mehrere Millionen Lebensversicherungsverträge. Bei Banken stünde hier ein System für die Abwicklung von Zahlungsverkehr oder Aktienhandel mit Millionen von Transaktionen pro Woche.
- Spezialisierte Verkaufsanwendungen und Anbindungen an Vertriebspartner.
- ERP-Anwendungen. Also typischerweise standardisierte Anwendungen für sekundäre Geschäftsprozesse wie Hauptbuchhaltung, Controlling, Personalverwaltung und Ähnliches. Solche Anwendungen kann man von ERP-Herstellern, wie zum Beispiel SAP, Baan, Peoplesoft oder Oracle, zukaufen
- Data-Warehouse (DWH)- und Statistikanwendungen. Diese entstehen auf zugekaufter Basissoftware und decken den so genannten dispositiven Bereich ab. Für solche Lösungen haben sich Architekturstandards entwickelt. Siehe dazu zum Beispiel [Kimb+1998, Wiek1999].

Die Reihenfolge ist nicht zufällig so gewählt. Die Kernanwendungen waren meist zuerst vorhanden. Vielfach enthielten sie das, was heute unter den Kategorien ERP und DWH zugekauft wird, aber noch vor 25 Jahren ebenfalls selbst entwickelt wurde. Verkaufsanwendungen wurden später ab den 80er Jahren dazu gebaut, und zwar mit dem Aufkommen von mittlerer Datentechnik für dezentrale Einheiten, mit dem Aufkommen des PCs und mit der Verbreitung von Laptops.

[1] ETL ist ein Akronym für *extract, transform, load*. Man benutzt ETL-Tools, um Inhalte aus operativen Datenbanken zu extrahieren, diese zu transformieren und hinterher zum Beispiel in ein Data Warehouse zu laden. ETL Tools können oft auch für Datentransformationen im EAI-Bereich eingesetzt werden.

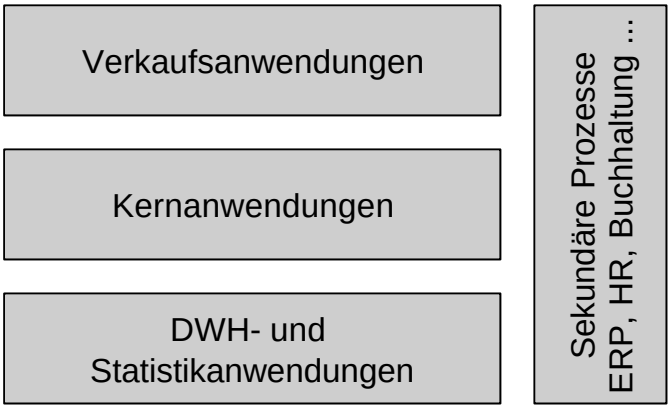


Abbildung 3: Oberste Sicht auf das Anwendungsportfolio einer Versicherung

Das Enterprise Nervous System, also die oben genannten Arten von technischer Software, wird zunehmend als Klebstoff und Schmiermittel benützt, um Applikationen zusammenzufügen, kommunizieren zu lassen und um Inhalte für Geschäftspartner verfügbar zu machen – zum Beispiel im Web.

In diesem Buch beschäftigen wir uns näher mit den ersten drei Arten von Bestandteilen eines ENS, nämlich Integration Brokern, Geschäftsprozessmanagementtools und Middleware zur Kommunikation. Webserver und Applikationsserver sind hier nicht unser Thema. Mit ETL-Tools werden wir uns nur am Rande befassen, sofern sie auch EAI bei der Datentransformation unterstützen können.

2.2 Trends, die den Bedarf nach EAI wecken

In diesem Abschnitt soll der Frage nachgegangen werden, warum der Markt und der Bedarf für EAI-Lösungen derzeit dramatisch wächst. Der Satz:

Das Internet wird für immer die Art verändern, wie wir Geschäfte machen,

ist zwar schon ein guter Einstieg, aber noch keine vollständige Erklärung, weil sich hinter dem Begriff eine große Menge von Technologien und Möglichkeiten verbergen, die man differenzierter betrachten muss.

2.2.1 Trend 1: Internet, B2B und neue Geschäftsprozesse

Der Internet-Hype der letzten Jahre vor dem Dotcom-Crash war geprägt durch so genannte B2C-Modelle, also Business-to-Customer-Modelle. Die Geschäftsmodelle, die dort primär implementiert wurden, waren (nach Musil et al. [Musil2000]) folgende:

?? **Verkehr erzeugen und verkaufen:** Auf diesem Geschäftsmodell basieren Suchmaschinen, Verzeichnisdienste und zum Beispiel kostenfreie E-Mail-Provider und ähnliche Anbieter. Der Betreiber eines Internetauftrittes versucht, möglichst viel Verkehr auf seine Seiten zu ziehen, den Kunden dort möglichst lange zu halten, und erzielt Einnahmen über Werbung.

?? **Kundenorientierung zum Beispiel im elektronischen Handel:** Unter diese Kategorie fallen die großen Internet-Versandhändler. Das Geschäftsmodell funktioniert dann am besten, wenn man beim Verkaufen sein Wissen über den Kunden ausnutzt. Wie solche Anbieter gut mit Customer Relationship Management (CRM) umgehen können, kann man zum Beispiel sehr gut studieren, wenn man Kunde von amazon.com ist. Die Lesevorschläge, die man dort bekommt, stellen einen echten Mehrwert gegenüber einem normalen Einzelhandel dar. Die Auswahl ist enorm groß und die Lieferung ist nicht langsamer als bei einer Bestellung im normalen Buchhandel.

?? **Prozesseffizient, also B2C-Marktplätze und Vermittlung von Geschäften:** Eine weitere Kategorie von Geschäftsmodellen ist die Vermittlung von Geschäften. Darunter fallen zum Beispiel elektronische Auktionshäuser, elektronische Vergleichsplattformen und sonstige Vermittlung von Waren und Dienstleistungen über das Netz. Diese Geschäftsmodelle funktionieren, weil sie effizientere Prozesse bieten als ohne den Einsatz des Internets zur Unterstützung.

Allen den obigen Modellen ist gemeinsam, dass sie nicht unbedingt Treiber für EAI-Anwendungen sind. Am ehesten kann noch aus den letzten beiden Modellen der Bedarf nach EAI-Lösungen entstehen, nämlich dann, wenn für die Veranlassung von Lieferungen, Verbindungen mit Geschäftspartnern aufgebaut werden müssen.

Einige der aus den obigen Modellen hervorgegangenen Firmen sind stabil und werden bestehen bleiben. Viele weniger solide Marktteilnehmer mit schwachen Geschäftsmodellen sind schon ausgeschieden. Übertragen auf Abbildung 4 befinden wir uns derzeit knapp hinter dem Punkt, der mit „The empire strikes back“ markiert ist.

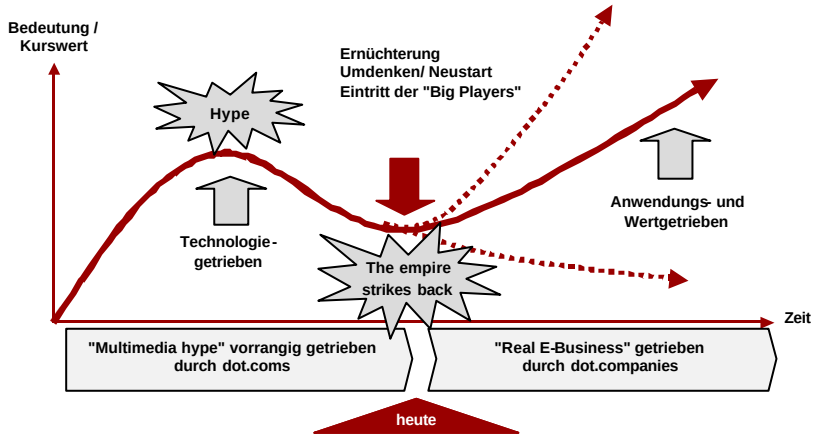


Abbildung 4: Hype-Kurve für E-Business und EAI

Dieser Punkt zeichnet sich dadurch aus, dass auch ältere, lange bestehende, konservative und etablierte Unternehmen anfangen, die Möglichkeiten des E-Business und des Internets zu erkennen und zu verwenden. Dort wird Internettechnologie vor allem für folgende Anwendungsfälle genutzt:

?? **Neugestaltung von Geschäftsprozessen:** Begriffe wie Business Process Reengineering gibt es schon länger und die „radikale Neugestaltung von Geschäftsprozessen“ hatte nicht immer einen guten Ruf. Trotzdem kann man in sehr vielen Unternehmen Kosteneffekte dadurch erzielen, dass man bestehende Geschäftsprozesse vereinfacht oder mit Hilfe von Technologieeinsatz komplett neu erfindet. Dabei wird man vor allem versuchen, Bearbeitungsschritte zu automatisieren

oder ganz wegfallen zu lassen und das 80/20-Prinzip anzuwenden. Also zum Beispiel 80% der Fälle möglichst automatisch abzuwickeln und menschliche Bearbeiter nur dort einzusetzen, wo es unvermeidbar ist.

- **B2B-Integration:** Eine Variante der Neugestaltung von Geschäftsprozessen ist die Business-to-Business-Integration. Hier werden die Wertketten von Unternehmen durch Kommunikationstechnik verbunden. Es bestehen Beschleunigungs- und damit Kosteneffekte. Zum Beispiel kann das Bestellwesen automatisiert werden.

- **Verlagerung von Service auf billigere Kanäle:** Zum Boom der Callcenter hat beigetragen, dass sie es erlauben, ähnlichen Service preiswerter zu erbringen als zum Beispiel mit einer flächendeckenden Service-Organisation. Selbstbedienung über das Internet ist noch mal preiswerter als Service über das Callcenter. Viele Unternehmen versuchen also möglichst viele Prozesse nach außen zu Kunden und Geschäftspartnern zu verlagern, um Kosten zu sparen.

- **B2E-Modelle:** Eine Variante der Verlagerung von Service auf billigere Kanäle sind auch Business-to-Employee-Portale. Statt Mitarbeiter durch andere Mitarbeiter bedienen zu lassen, werden die nötigen Informationsleistungen über elektronische Portale erbracht.

Alle diese Ansätze haben eines gemeinsam: Um sie zu implementieren, muss man alte und neue Anwendungen integrieren und existierende DV-Systeme für Nachrichten aus dem Internet zugänglich machen. Dies sind also primäre Einsatzfelder für EAI-Technologien.

2.2.2 Trend 2: ERP-Lösungen und Komponenten

Ein weiterer Trend am Markt hat nur sekundär etwas mit dem Internet zu tun. So genannte ERP-Lösungen sind große fertige Softwarepakete, die von den meisten größeren Unternehmen heute für sekundäre Geschäftsprozesse wie Rechnungswesen, Materialwirtschaft und vieles mehr eingesetzt werden. Prominentester Vertreter solcher Systeme ist SAP.

Diese Systeme müssen natürlich mit den proprietären und zum Teil wettbewerbskritischen Anwendungen der Unternehmen verbunden werden.

Der Trend geht dahin, dass immer mehr Software in den Unternehmen im Einsatz ist und davon prozentual immer weniger selbst programmiert wird. Das heißt, dass sich die Aufgabenschwerpunkte von EDV-Mitarbeitern verschieben, und zwar von der Programmierung proprietärer Lösungen hin zur Integration fertiger Lösungen zu den Geschäftsprozessen, die das Unternehmen benötigt. Die EDV-Mitarbeiter der ERP-Anwenderunternehmen werden also nicht arbeitslos, sondern haben dann vor allem zwei Arten von Aufgaben:

- Customizing der zugekauften ERP-Anwendungen

- und Integration dieser Anwendungen mit den eigenen Anwendungen mittels GlueWare, wie zum Beispiel EAI-Tools.

Weitere Entwicklungen verstärken diesen Trend. Die Vorstellungen aus den 80er und 90er Jahren, dass ein Unternehmen ein möglichst integriertes Gesamtsystem in einer möglichst homogenen Technologie haben sollte, befinden sich auf dem Rückzug, weil sie sich als nicht praktikabel erwiesen haben.

?? Zum einen sind die Technologiezyklen heute so kurz, dass es kaum noch ein Unternehmen schaffen wird, seine komplette Software innerhalb eines Technologiezyklus von 2-3 Jahren neu zu schreiben. Die typische Lebensdauer von geschäftskritischen Systemen liegt eher bei 20-30 Jahren und es dauert zum Beispiel in einer Versicherung um die 7-10 Jahre, ein Anwendungsportfolio einmal komplett zu erneuern. Daher müssen sich die Daten- und Architekturpolizisten mit dem Gedanken vertraut machen, es mit mehr als einer Technologiegeneration zu tun zu haben. Diese Systeme müssen integriert werden. Mithin handelt es sich wieder um einen potenziellen Anwendungsfall für EAI-Technologien.

?? Zum anderen ist es gar nicht immer wirklich gut, ein voll integriertes System zu haben. Lose gekoppelte Komponenten können wesentlich leichter zu betreiben sein. Siehe dazu auch das Beispiel in Abschnitt 8.1.4, in dem es um die Vorteile „schlecht" integrierter Systeme geht.

Schließlich gibt es noch den Trend „Componentware". Wenn man Architekturen wie J2EE oder .NET als Componentware bezeichnet, wird Componentware eine Rolle spielen. Bei der Integration von existierenden, alten Systemen sind die Versprechungen des Component-Hypes allerdings nicht wahr geworden. Der wirklich große Component-Hype flaut daher schon wieder ab. Für EAI spielt Componentware nicht DIE große Rolle. Trotzdem werden wir auch die Integration über Komponenten in diesem Buch mit diskutieren. Siehe dazu Abschnitt 4.1.4.3.

2.2.3 Trend 3: Fusionen

Eine weitere Strömung, die die Integration von Software zu einer häufig geübten Sportart macht, die jeder EDV-Profi beherrschen muss, sind die Unternehmenszusammenschlüsse und Fusionen. Sie kommen derzeit in der Finanzindustrie immer häufiger vor und haben teilweise gigantischen Umfang. Andere Industrien nutzen ebenfalls Economies of Scale.

Das Endziel bei solchen Fusionen wird es meist sein, das EDV-System eines der Fusionspartner zu behalten und die Bestände des anderen Partners in das beibehaltene System zu migrieren. Dieses Vorgehen ist jedenfalls aus guten Gründen bei Fusionen im Finanzdienstleistungssektor sehr gebräuchlich, weil die Systeme meist skalierbar genug sind, um das Geschäft eines der Fusionspartner noch mit aufzunehmen. Man kann sich damit sehr oft doppelte Wartung ersparen[2]. Bis man so weit ist, vergehen allerdings meist 1,5 bis 3 Jahre, in denen man zum Beispiel gemeinsame Sichten auf Datenbestände beider Unternehmen benötigt. Dies kann dann wieder ein Anwendungsfall für EAI-Technologien sein.

2.3 Einige Anwendungsfälle für EAI-Lösungen

In diesem Abschnitt lernen Sie vier typische Anwendungsfelder für EAI näher kennen. Die Beispiele stammen großteils aus dem Bereich der Finanzindustrie, denn dieses Buch hat,

[2] Die Begründung dafür ist länger und nicht Thema dieses Buches und kann bei Interesse in [Kell2001] nachgelesen werden.

was die Beispiele anbelangt, einen Schwerpunkt auf Versicherungen und Banken. Der Analogieschluss zu anderen Industrien ist meist nicht sehr schwer.

Andere Fälle aus eher industriellen und sonstigen Umfeldern werden dann danach noch in Kürze vorgestellt.

2.3.1 Multichannel-Architekturen

Um zu verstehen, warum man so genannte Multichannel- (oder auch Mehrkanal-) Architekturen unterstützen muss und welche Softwarewerkzeuge man dafür benötigt, ist ein kleiner Blick in die jüngere Vergangenheit von Versicherungen hilfreich.

Noch vor etwa 10 Jahren sah die Anwendungslandschaft eines Versicherungsunternehmens in etwa so aus, wie in Abbildung 5 dargestellt. Data-Warehouse-Anwendungen und ERP-Anwendungen waren oft noch in die Kernsysteme integriert. Uns interessiert für die Diskussion hier auch nur die Beziehung zwischen Verkaufssystemen (Front-Office) und Kernsystemen (Back-Office). Auch wenn im Vertrieb Laptops vorhanden waren, so erfolgte dennoch ein Großteil des Informationsflusses (Pfeil) zwischen Verkäufer und Back-Office über Papier.

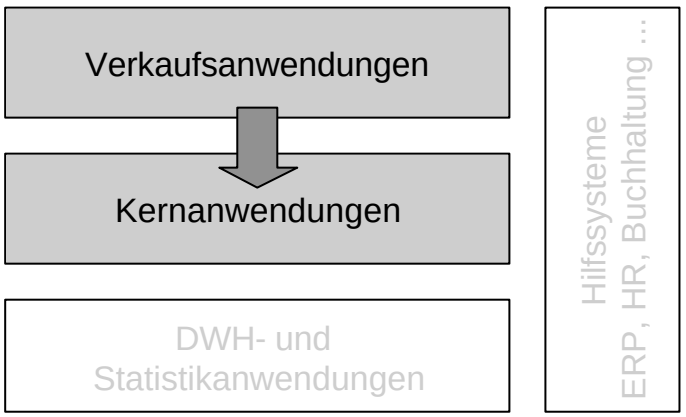


Abbildung 5: Anwendungsportfolio grob – frühe 90er Jahre. Damals erfolgte ein Großteil des Informationsflusses zwischen Verkaufsanwendungen und Back-Office (Kernanwendungen) noch über Papier.

Schon damals existierte mehr als nur ein Vertriebskanal. So gab und gibt es als Vertriebskanäle den von Versicherungsunternehmen angestellten Außendienst, Agenten, Makler und diverse Partnerorganisationen. Auch Kunden benötigen Auskünfte und direkte Kontakte.

Mit dem Internet bieten sich nun mehrere Möglichkeiten der Verbilligung von Geschäftsprozessen an:

- Durch das Aufkommen von Callcentern und Internetdialogen konnte man darangehen, Auskünfte an Kunden auf Kanäle zu verlegen, die billiger sind als die Auskunft durch einen Außendienstmitarbeiter, Makler oder Innendienstmitarbeiter am Telefon.

- Die Außendienstkanäle, die nicht Teil des Unternehmens sind (Makler, Agenten, Partnerorganisationen), können ebenfalls mit Selbstbedienung und automatisiertem Datenaustausch besser und billiger angebunden werden, als das über telefonische Auskünfte durch einen Innendienst bisher der Fall war.

Diese beiden Entwicklungen hatten und haben Auswirkungen auf die Architektur der Informationssysteme. Die Welt vor dem Back-Office-System wird um einiges vielfältiger und komplexer.

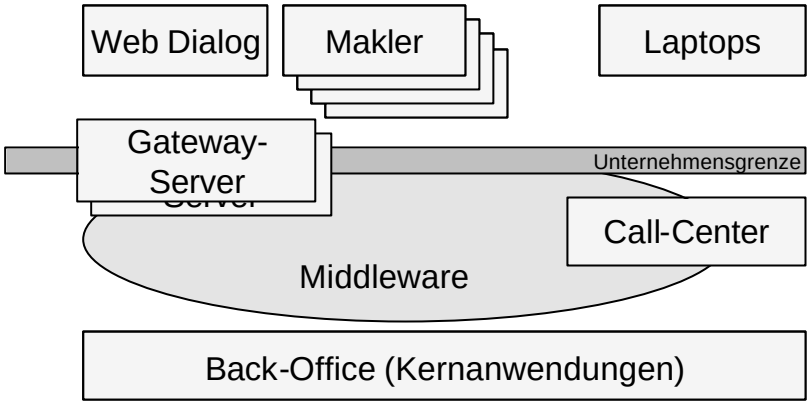


Abbildung 6: Typische Front-Office-Landschaft einer Versicherung

Abbildung 6 zeigt dies. Vor der Unternehmensgrenze gibt es mehr als eine Anwendung, die Geschäftspartnern zur Verfügung steht. Für Kunden sind Web-Dialoge als Anwendungen zur Selbstbedienung vorhanden. Für Makler und Agenten gibt es entweder spezialisierte Anwendungen als Web-Anwendungen oder Datenaustausch über normierte Formate. Der eigene Außendienst arbeitet nach wie vor weiter mit Laptops oder demnächst permanent online mit Smartphones und Laptops. Das Callcenter steht als weiterer Kanal für die Bedienung von Kunden und Vertriebspartnern zur Verfügung.

Die Unternehmensgrenze wird oft durch eine Firewallarchitektur dargestellt, durch die das Unternehmensnetz vom Internet abgetrennt und gegen Angriffe abgesichert wird. Gateway-Server übernehmen die Kommunikation nach außen.

Man kann dieses Bild sicher noch detaillierter darstellen. Wichtig ist jedoch vor allem der Aspekt der Middleware. Im Endeffekt werden bei der oben abgebildeten Multichannel-Architektur aus immer denselben Backend-Anwendungen die verschiedensten Frontend-Anwendungen bedient, allerdings ist die Aufbereitung jedes Mal eine geringfügig andere.

?? Die Selbstbedienungs-Internetanwendung für Kunden lässt nur wenige, einfache Geschäftsfälle zu.

?? Im Kundenservicecenter (Callcenter) gibt es schon einen erweiterten Zugriff auf die Kernsysteme, zum Beispiel über einen Intranetdialog. Hier werden mehr Geschäftsfälle angeboten, zum Beispiel Änderungen von Verträgen.

?? Und die Offline-Laptops der Außendienstmitarbeiter werden über eine Datenversorgung bedient, die den Außendienst mit wichtigen Kunden- und Vertragsdaten beliefert. Es ist weiter damit zu rechnen, dass auch der eigene Außendienst Onlinezugriffe erhalten wird, wenn mobile Datenkommunikation so billig ist, dass das kostenmäßig verträglich ist. Es entsteht dann ein weiterer Onlinekanal.

?? Die gleichen Vertragsinhalte werden auch für Makler entweder im Dialog oder als Versorgungspakete für deren eigene Datenverarbeitung aufbereitet.

2.3.1.1 N:1-Multichannel-Architekturen

Wir haben es also hier mit Architekturen zu tun, bei denen eine Art von Kunde über mehrere verschiedene Kanäle bedient wird. Allerdings werden die Kunden immer wieder aus demselben Backend-System mit grundsätzlich denselben Informationen beliefert. Abbildung 7 zeigt eine solche Situation. Ein Kunde kann hier über drei Kanäle bedient werden.

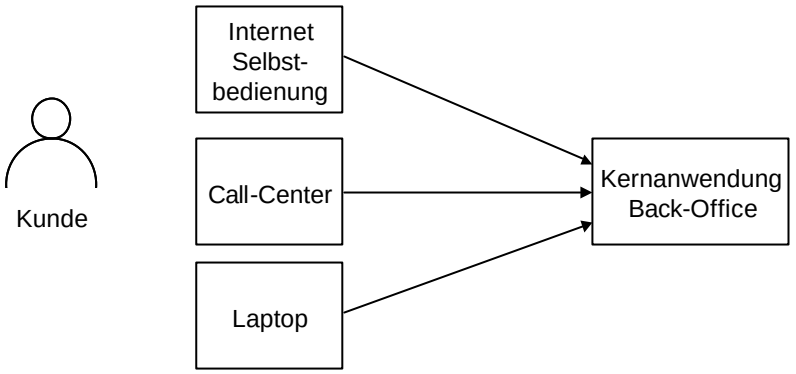


Abbildung 7: N:1-Multichannel-Situation

Abbildung 7 trägt aber bewusst noch nicht den Titel „Architektur“. Eine vernünftige Architektur muss es vermeiden, dass für jeden Kanal eine neue, individuelle Schnittstelle entsteht. Würde man ohne Architekturkonzept einen Kanal nach dem anderen an das existierende Kernsystem (Back-Office-System) andocken, könnte es zu einer Situation kommen, in der es für jeden Kanal eine eigene Schnittstelle für dieselbe oder sehr ähnliche Funktionalität gibt. Das wäre zu teuer und ist daher zu vermeiden.

Ein vernünftiger Einsatz von Middleware, und speziell von EAI-Middleware, erlaubt es hier, die Erstellungs- und Wartungskosten zu senken. Was für einen solchen Ansatz benötigt wird, ist abstrakt in Abbildung 8 gezeigt.

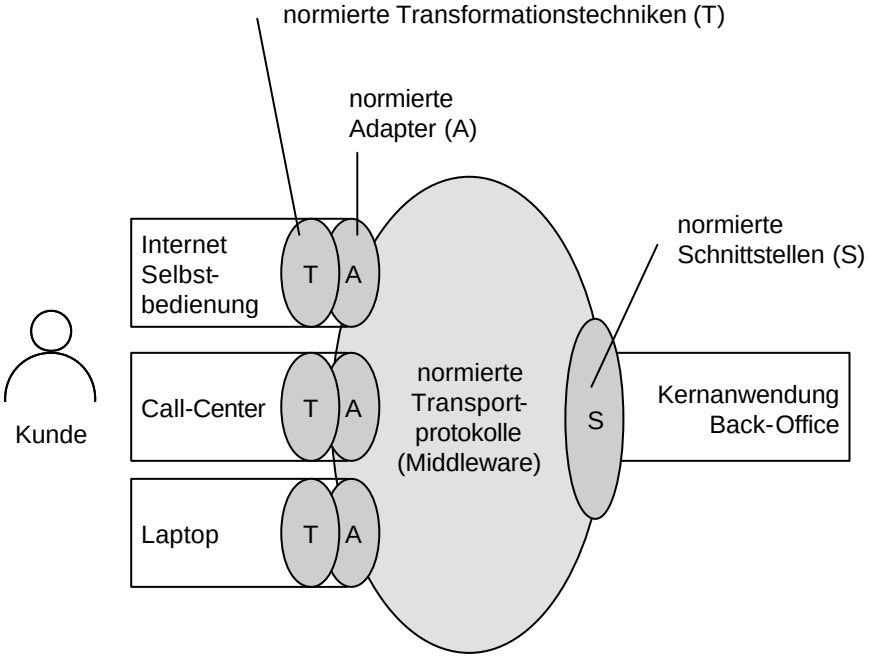


Abbildung 8: Einsatz von EAI-Technologien im N:1-Multichannel-Szenario

Einsparungen kann man durch folgende Vereinheitlichungen erzielen:

- Zunächst macht es Sinn, wenn es möglichst nur eine Art der technischen Kommunikation gibt, damit nicht jeder Kanal über eine eigene Kommunikationsarchitektur mit der Kernanwendung kommuniziert.

- Dann sollte man die Schnittstelle des Kernsystems normieren. Man sollte versuchen, einfach aufzurufende Schnittstellen zur Verfügung zu stellen, die von mehr als einem Kanal genutzt werden können. Wer weiß, dass die Schnittstellen alter Hostsysteme vor allem für Onlinetransaktionen designt sind, der weiß, dass die Normierung von Schnittstellen alles andere als ein triviales Unterfangen ist. Damit beschäftigt sich daher im Speziellen Abschnitt 4.1.4.

- Auch die Art, wie Frontend-Systeme auf die Middleware zugreifen, sollte weitgehend genormt sein. Wir sprechen hier im Folgenden von so genannten Clips. Siehe dazu speziell noch Abschnitt 5.1.2.

- Ebenso sollte auch die Art, wie aus einer möglichst gleichen Nachricht mehrere leicht unterschiedliche Darstellungen erzeugt werden, in einem Framework normiert sein. Die Werkzeuglandschaften rund um XML (wie zum Beispiel XSLT) bieten hierfür reichliche Möglichkeiten.

Wenn Sie EAI-Integrationsserver wie Vitria, TIBCO oder andere schon kennen sollten, werden Sie bemerken, dass man von den angebotenen Fähigkeiten solcher Tools für das hier dargestellte Szenario nur einen Bruchteil benötigt. Dies ist einer der Gründe, warum solche Tools nicht sehr verbreitet sind, wenn ein Kunde nur den oben dargestellten Anwendungsfall hat.

Speziell das so genannte Routing ist hier trivial. Unter Routing versteht man die Sicherstellung, dass ein Aufruf den richtigen Server erreicht. Wenn alle Aufrufe immer von demselben Backend bedient werden, ist das Routen der Aufrufe logischerweise trivial.

Wenden wir uns also einer etwas komplexeren Familie von Anwendungsszenarien zu.

2.3.1.2 N:M-Multichannel-Architekturen

Bisher haben wir uns mit dem Szenario beschäftigt, bei dem im Wesentlichen ein Kernsystem (Back-Office-System) die Anforderungen aus den Kanälen abarbeitet. In den meisten Versicherungsunternehmen (und auch Banken) hat man es allerdings mit deutlich mehr als einem Server zu tun, der die nötigen Informationen liefert, um eine Gesamtsicht für einen Kunden zu erzeugen.

- Bei einer Versicherung sind typischerweise oft das Sachversicherungssystem und das Lebensversicherungssystem getrennt. Häufig kann es auch mehr als ein Lebensversicherungs- oder Sachversicherungssystem geben.

- In einer Bank gibt es typischerweise mindestens ein System für laufende Konten und eines für Wertpapierhandel.

Oft liegen alle diese Systeme auf einer Maschine. Manchmal ist aber auch das schon nicht mehr der Fall. Spätestens dann aber, wenn ein Unternehmen auch Marktleistungen anderer Anbieter offeriert, ist das N:1-Multichannel-Szenario durchbrochen. Sowohl Banken als auch Versicherungen verkaufen Produkte des jeweils anderen Bereiches der Finanzindustrie mit und versuchen Allfinanzkonzepte zu implementieren.

Damit ergibt sich ein Trend zu mehr Kanälen und gleichzeitig mehr Servern. Diese Server für Kernapplikationen (Back-Office-Systeme) werden wir im Folgenden Produktfabriken nennen. Wir haben es durch Ideen zur Kostensenkung mit mehreren Kanälen zu tun. Durch die horizontale Integration von neuen Produkten werden gleichzeitig im Bereich der Back-Office-Systeme mehrere Produktfabriken angesprochen. Um zum Beispiel als Versicherung ein Allfinanzangebot zur Verfügung stellen zu können, benötigt man noch mehr Server als Versicherungsvertragssysteme (siehe rechte Seite Abbildung 9), nämlich Produktfabriken für Bankprodukte.

Abbildung 9 zeigt, dass man, um zum Beispiel ein einheitliches Web-Interface für einen Kunden anzubieten, auf Versicherungsvertragssysteme zugreifen muss oder nun auch zum Beispiel auf eine Girokonto-Maschine.

Gleichzeitig müssen die Informationen der Sachversicherungsmaschine (Vertragssystem) in mehr als einer Oberfläche angezeigt werden.

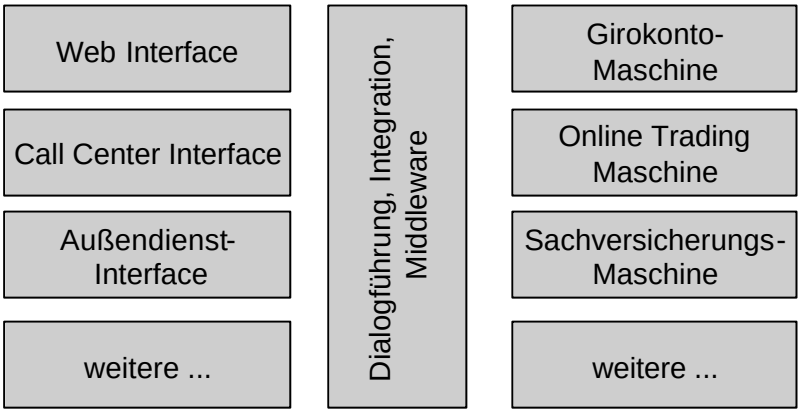


Abbildung 9: Multichannel-Architektur und integrierende Middleware

Solche Probleme geht man typischerweise wieder mit EAI (Enterprise Application Integration)-Lösungen an.

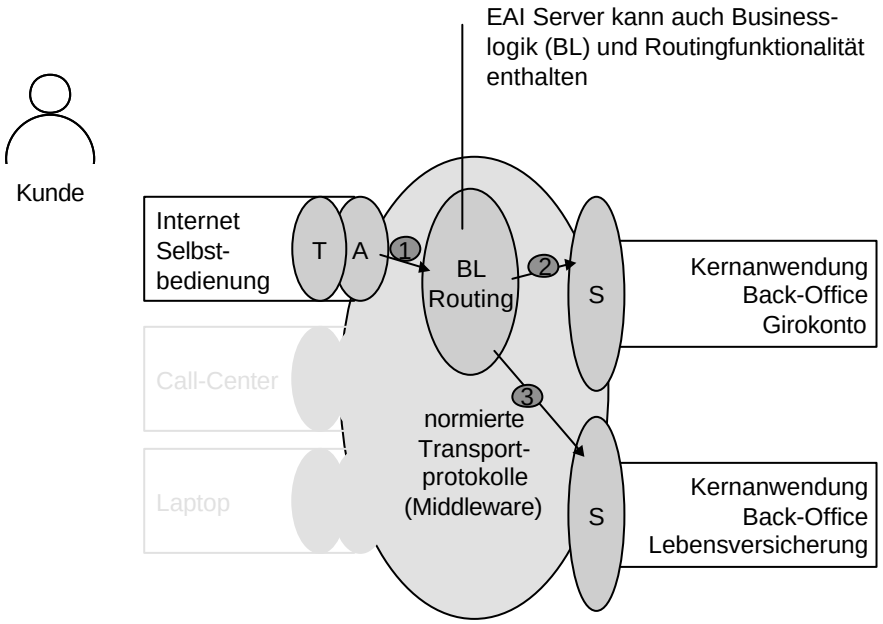


Abbildung 10:Einsatz von EAI-Technologien im N:M-Multichannel-Szenario

Abbildung 10 ist geringfügig komplexer als Abbildung 8. Als zusätzliche Funktionalitäten sollte die Middleware praktischerweise noch Folgendes beherrschen:

- Es sollte möglich sein, mit Hilfe der Business-Logik Dienste auf Ebene der Middleware zu definieren, die es in keiner der angeschlossenen Produktfabriken schon gibt. Zum Beispiel könnte sich der „Netto-Vermögenswert“ eines Kunden aus dem Wert seiner Girokonten und Lebensversicherungen zusammensetzen. Es macht keinen Sinn, wenn jeder Channel diese Formel und dieses Wissen wieder neu implementiert. Besser ist es, wenn dies zum Beispiel in einem EAI-Server implementiert wird. BL steht dabei in Abbildung 10 für Business Logic, also Geschäftsfunktionalität.

- Weiter macht es Sinn, zumindest elementare Routing-Fähigkeiten in der Middleware zu haben, da man nicht jede Menge Software durchsehen möchte, wenn sich ein Produktgeber oder auch nur die Adresse der Produktfabriken im Netz ändert.

2.3.1.3 Lage eines EAI-Servers in einer N:M-Multichannel-Architektur

In Abbildung 8 und Abbildung 10 wurde die EAI-Funktionalität als Wolke zwischen den Kanalanwendungen (Clients, Frontends) und den Produktfabriken gezeichnet. Eine solche Wolke gibt es weder als „Stück Software“ noch wäre dies eine besonders befriedigende Architekturskizze. Wir wollen daher kurz beschreiben, mit welcher Art von Servern man es dann in der Realität zu tun hat und welche Verantwortlichkeiten diese haben. Nicht jeder Server muss auf einer eigenen Maschine laufen, er könnte es aber.

Für die Beispiele und Arten von Servern und Software werden wir im Folgenden die Terminologie von J2EE (Java 2 Enterprise Edition) verwenden. Analoge Architekturen kann man auch für Microsofts .NET-Plattform aufbauen, allerdings mit einer leicht unterschiedlichen Terminologie.

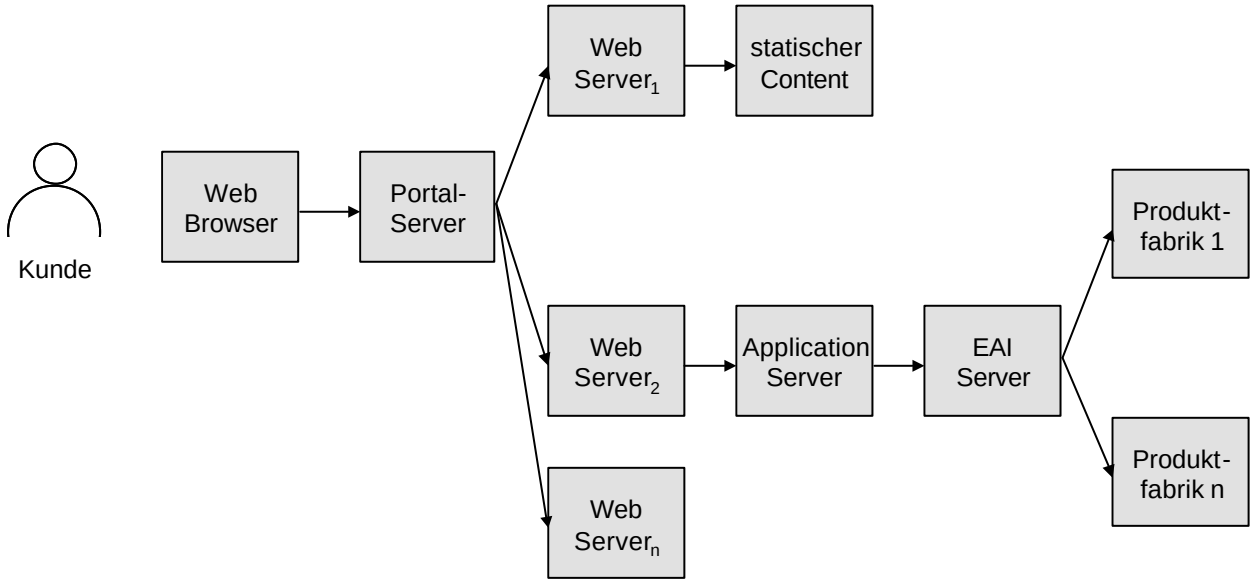


Abbildung 11: Vom Browser zur Produktfabrik

Die in Abbildung 11 dargestellten Server haben die folgenden Verantwortlichkeiten:

Server	Verantwortung
Portalserver	Der Portalserver bietet dem Benutzer einen Einstieg (auch personalisiert), die Authentifizierung und bindet statische Web-Inhalte an.
Webserver	Die Webserver stehen hier für Server für statische oder dynamische Inhalte. Die Webserver können auch Komponenten enthalten, die Benutzerinteraktionen auf der Basis von Eingaben des Benutzers steuern. Das geht dann über die reine statische Web-Interaktion hinaus. Die Formulare einer Anwendung für Überweisungen würden zum Beispiel von einem Webserver gerendert werden. Dieser würde auch einen Session-Kontext für den Benutzer mindestens während seiner Überweisung halten. Im J2EE-Modell würden auf dem Webserver vor allem die JSPs (Java Server Pages) laufen, die die dynamischen Webseiten erzeugen. Es ist klar, dass hinter einem Portalserver mehrere Applikationen und damit beliebig viele Webserver in Frames laufen können.
Application Server	Ein Application Server stellt Geschäftsfunktionalität zur Verfügung. Der Webserver kann entweder einen Application Server benutzen, wenn der Kunde komplexere Objekte zu sehen bekommt. In einem solchen Fall würde auf dem Application Server zum Beispiel ein EJB-Container laufen. Oder es können auch Proxies (Java-Servlets) für andere Funktionalitäten dort laufen.
EAI-Server	Der EAI-Server übernimmt Business Logic und Routing sowie unter Umständen auch Schnittstellennormalisierung für nachgelagerte Back-Office-Systeme. Die genaue Aufgabenverteilung zwischen EAI-Server und Back-Office-System wird in

5.2 Fallstudie 2: Vitria Businessware

Als zweiten Fall eines EAI-Integrationsservers wollen wir Vitria Businessware betrachten. Vitria gehört zu den Herstellern von EAI-Produkten, die einen großen und kompletten Funktionsumfang anbieten und außerdem ein stabiles produktionsreifes Produkt. In einen magischen Quadranten übersetzt bedeutet das, dass man das Produkt im Quadranten der Marktführer (siehe Abbildung 86) finden wird.

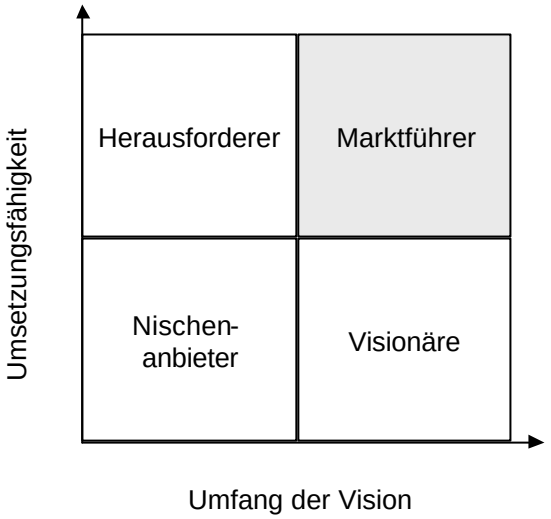


Abbildung 86: Magischer Quadrant[11]

Wir betrachten daher Vitria hier stellvertretend für ein modernes EAI-Produkt mit vollem Funktionsumfang. Vitria eignet sich vor allem auch gut, um das Zusammenwachsen von EAI und Geschäftsprozessmanagement zu demonstrieren.

5.2.1 Architektur von Vitria Businessware

Gute Architekturen erkennt man immer daran, dass man die Grundidee dazu sehr schnell begreift und diese so mächtig ist, dass man damit alle wesentlichen Probleme des Bereiches einfach lösen kann, für den die Architektur entworfen wurde. Architekturen, die man lange und umständlich erklären muss, gehören selten zu den wirklichen Spitzenprodukten.

5.2.1.1 Grundmodell der Vitria-Architektur

Das Grundmodell von Vitria ist sehr einfach. Wir beginnen hier mit der Kommunikationsschicht der Referenzarchitektur. Vitria basiert auf einem asynchronen Nachrichtenmodell nach dem Publish/Subscribe-Modell.

---

[11] Der EAI-Markt ist sehr groß. Gartner Research betrachtet in ihren Studien eine zweistellige Anzahl von EAI Herstellern. Von denen bewegen sich jeweils ca. 5 im Feld der Markführer. Vitria ist einer von diesen Herstellern. Aus verständlichen Gründen verbietet es Gartner Research, dass man „magic quadrants“ in anderen Publikationen abdruckt, da veraltete Information zu einer Rufschädigung für Gartner Research und außerdem zu einem verringerten Absatz von Gartner Research führen würden. Wenn man Investitionsentscheidungen über mehrere Millionen Euro trifft, ist das Geld in aktuelle Gartner-Marktinformation sicher gut angelegt. Daher erklären wir hier das Grundmodell des magischen Quadranten und verraten auch, dass Vitria Ende 2001 zu den Marktführern gehörte, empfehlen aber, vor einer Entscheidung aktuelles Gartner-Material zu Rate zu ziehen.

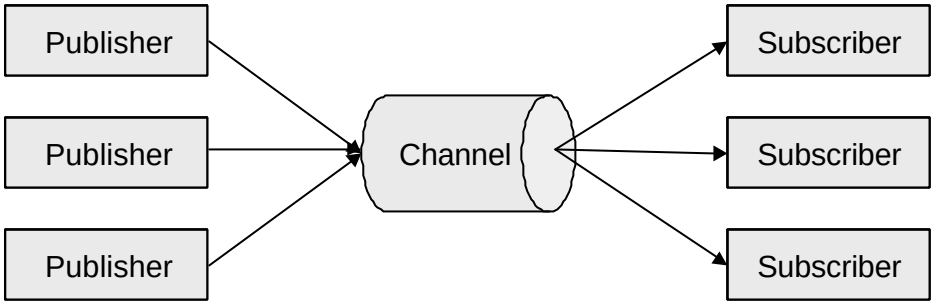


Abbildung 87: Vitria-Kommunikationsmodell

Die Begrifflichkeit ist etwas anders, als sie allgemein verwendet wird. Dazu kommt hier bei einem deutschen Buch noch das Übersetzungsproblem. Wir werden also zunächst die Vitria-Begriffe anhand derjenigen anderen Begriffe erklären, die in diesem Buch bisher verwendet wurden.

Tabelle 4: Übersetzung der Vitria-Terminologie

Vitria-Terminologie	Übersetzt in die Terminologie dieses Buches	Erläuterungen
Publisher	**Sender** einer Nachricht	
Subscriber	**Empfänger** der Nachricht, Abonnent einer zu definierenden Art von Nachrichten	
Channel	Nachrichtenkanal, **Kanal**	Der Begriff ist in Publish/Subscribe-Modellen nicht zwingend zu verwenden, aber nicht unpraktisch. In der Begriffswelt des Publish/Subscribe-Entwurfsmusters (deutsch Beobachter oder auch Abonnement) registriert sich ein Beobachter/Abonnent für eine Art von Ereignissen.
Event	**Nachricht**	Dies ist von der Begrifflichkeit her eher gewöhnungsbedürftig. Normalerweise würde man in Englisch den

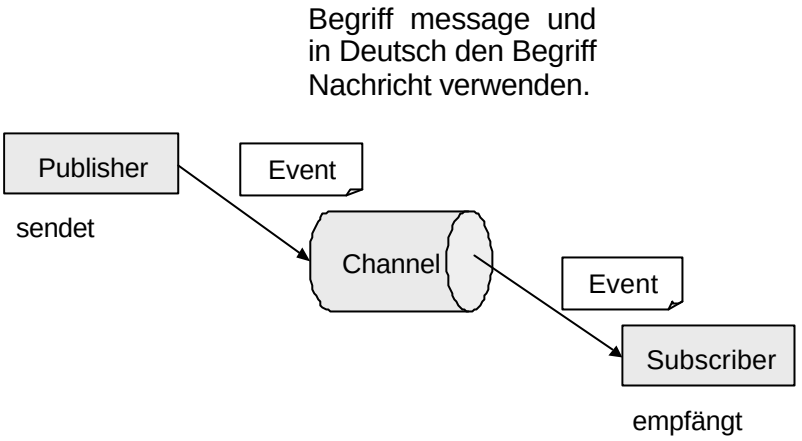


Abbildung 88: Nachrichten bei Vitria

Abbildung 88 zeigt anhand einer anderen Sicht die Begriffswelt von Vitria. Sender senden ihre Nachrichten (Events) an einen Kanal (Channel), ohne zu wissen, wer die Nachrichten empfängt oder verarbeitet.

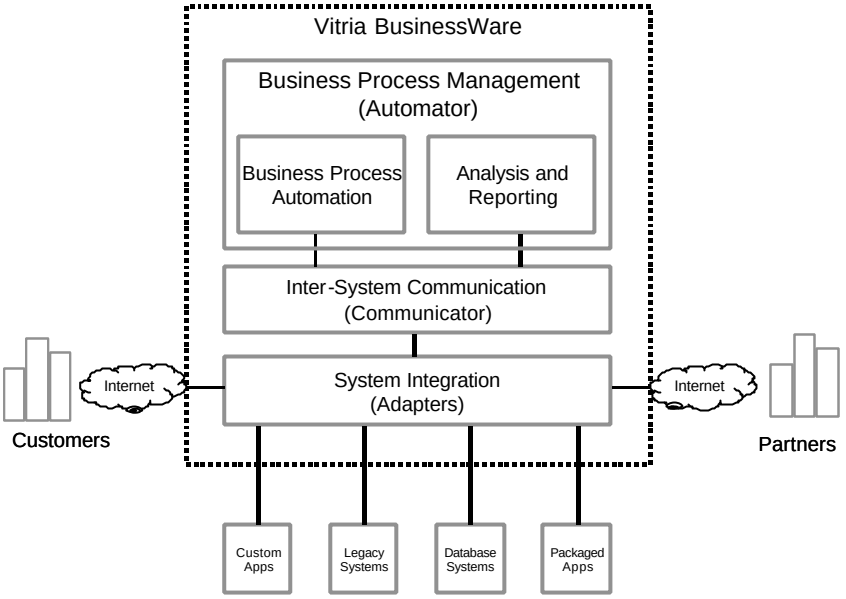
Ein Empfänger (Subscriber) kann einen Nachrichtenkanal abonnieren und bekommt alle Nachrichten zugestellt, die von Sendern auf diesem Kanal gesendet wurden. Ein Empfänger kann natürlich beliebig viele Kanäle abonnieren und ein Sender kann auf beliebig vielen Kanälen senden.

Da es sich bei dem Grundmodell von Vitria um ein asynchrones Publish/Subscribe-Modell handelt, kann man damit theoretisch auch synchrone Kommunikation abbilden sowie alle Formen asynchroner Kommunikation, die in Abschnitt 3.1.2 gelistet wurden. Es handelt sich hier also um ein einfaches und zugleich mächtiges Konzept.

5.2.1.2 Architektonischer Aufbau

Mit der obigen Diskussion ist noch nichts darüber gesagt, wie eine Vitria-Instanz mit der Außenwelt kommunizieren kann. Der Gesamtaufbau der Architektur wird in Abbildung 89 gezeigt.

Nachrichten, die die Sphäre von Vitria verlassen, werden durch Adapter der Schicht „System Integration“ empfangen oder gesendet. So wird zum Beispiel eine E-Mail durch einen Mail? Vitria-Adapter in einen Vitria-Event umgewandelt. Normalerweise haben solche Adapter eine Richtung. So gibt es sowohl einen Mail? Vitria-Adapter als auch einen Vitria? Mail-Adapter.



5.2.1.3 Abbildung 89: Vitria-Architekturüberblick und Produkte CORBA-Basierung

Die Definition der Nachrichtenformate (Events) geschieht in Vitria in CORBA-IDL. Der anwendende Softwarearchitekt hat damit die Wahl, ob er mit kompilierten Schnittstellen arbeiten möchte – nämlich dann, wenn alle Inhalte der Nachrichten im Interface bekannt gemacht und explizit typisiert werden – oder ob er eher XML-Stil verwenden möchte, also nur Kopfdaten im Interface der Nachrichten bekannt macht und den eigentlichen Inhalt in einem XML-Körper der Nachricht überträgt. Beides hat Vor- und Nachteile, die in anderen Kapiteln schon diskutiert wurden.

Hervorzuheben ist noch, dass Vitria einfache bijektive Konvertierung von CORBA-IDL in XML anbietet. Damit können Adapter auf einfache Weise eingehende XML-Nachrichten auf die interne, in CORBA-IDL definierte Schnittstelle umsetzen.

Bis hier ist Vitria ein normaler Integration Broker. Interessant wird das Konzept durch die Einbeziehung von Geschäftsprozessen.

5.2.1.4 Vitrias Sicht auf Geschäftsprozesse / Workflow

Vitria hat eine etwas andere Sicht auf die Steuerung von Geschäftsprozessen, als konventionelle Workflow-Manager das haben. Konventionelle Workflow-Manager gehen davon aus, dass ein Benutzer über die Aktivitäten, die er auslöst, aktiv die zu integrierenden Systeme anstößt, sich vom Workflow-Manager zwar durch Prozesse führen lässt, aber ansonsten der treibende Faktor im Geschäftsprozess ist.

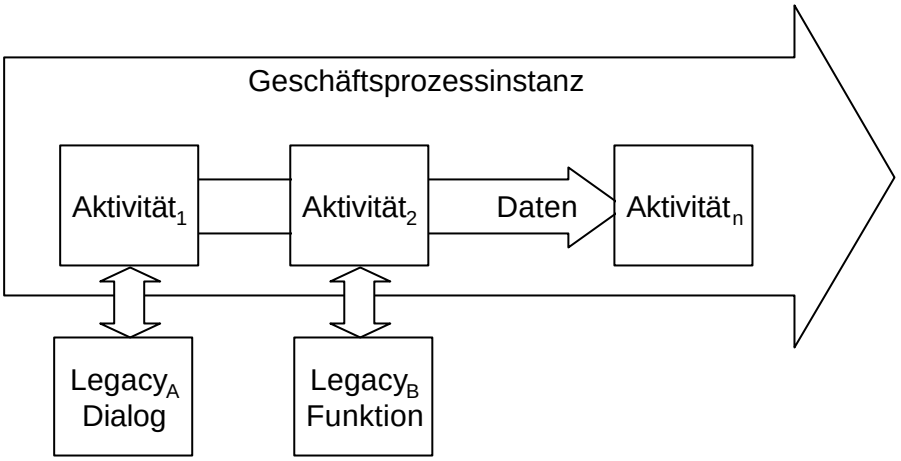


Abbildung 90: Konventionelle Sicht – Geschäftsprozess treibt Anwendungen

Diese Sicht ist in Abbildung 90 schematisch dargestellt. Der Prozess besteht aus einer Serie von Aktivitäten, die der Benutzer nacheinander ausführt. Der Workflow-Manager startet eine Aktivität und die Aktivität startet meist eines der zu integrierenden Dialogsysteme.

Die Sicht von Vitria ist eine etwas andere. Dabei ist der menschliche Bearbeiter der eigenen Firma nur einer von vielen Agenten, die etwas tun. Er ist gedanklich gleichberechtigt zu einem EDV-System eines Geschäftspartners oder irgendeinem anderen EDV-System.

Das Modell von Vitria bildet Geschäftsprozesse als Folge von Zuständen ab. Das Zustandssystem nimmt immer dann einen neuen Zustand an, wenn ein Ereignis eintrifft. Die Daten des Zustandes werden dann verarbeitet und verändern den Gesamtzustand des Geschäftsprozesses, danach legt sich der Vitria-Prozess wieder schlafen und wartet auf das nächste Ereignis. In Abbildung 91 ist ein Ausschnitt aus einem Prozess dargestellt, der in der Vitria-Philosophie modelliert ist.

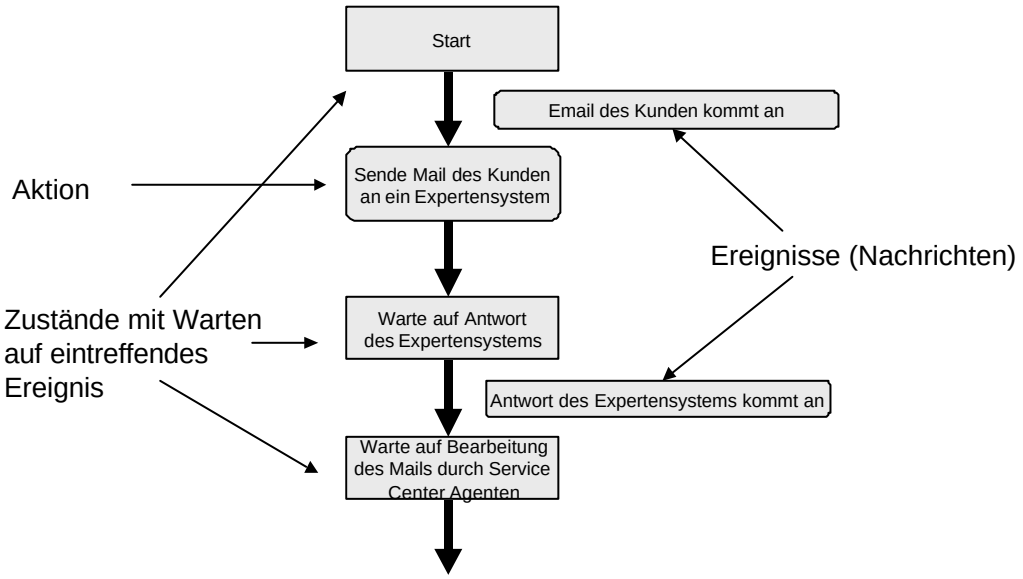


Abbildung 91: Prozess (analog zum Vitria-Modell)

Der Prozess wird durch eine E-Mail gestartet, die durch einen Mail? Vitria-Adapter in das System gelangt und in ein Ereignis (eine Nachricht) des Typs „E-Mail des Kunden kommt an“ umgewandelt worden sein muss. Die Frage ist natürlich, wie man die richtigen Mails

herausfiltert. Viele Unternehmen machen das, indem sie spezielle anonyme Adressen einrichten, wie service@mycompany.com.

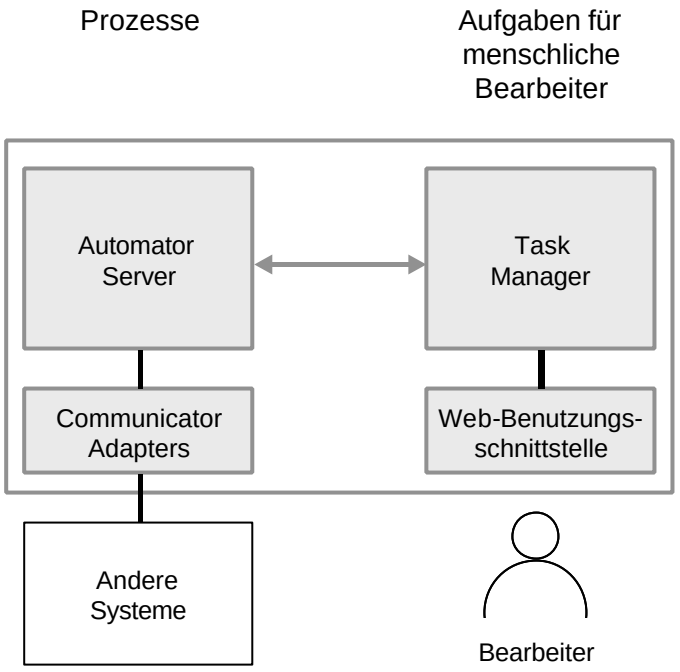


Abbildung 92: Taskmanager zur Abstraktion des Bearbeiterverhaltens in Events

Die ausgelöste Aktion besteht darin, dass die E-Mail an ein Expertensystem zur Bearbeitung weitergeleitet wird. Es wird also wieder ein Vitria-Event erzeugt, der den Mailtext als Nachrichtenkörper beinhaltet. Dies geschieht über einen vereinbarten Kanal. Das Expertensystem ist Abonnent dieses Kanals, empfängt die Mail, versucht Antworten darauf zu generieren und sendet einen Event „Antwort des Expertensystems kommt an“ an den Geschäftsprozess zurück. Es ist natürlich kein triviales, aber lösbares Problem, die richtige Geschäftsprozessinstanz zu dem Event zu finden.

Dann generiert der Geschäftsprozess eine Aktivität für einen Bearbeiter und legt sich so lange schlafen, bis die Aktivität abgearbeitet wurde. Dies wird von dem so genannten Taskmanager dem Geschäftsprozess wieder durch einen Event signalisiert.

Dieser Ansatz unterscheidet sich in zwei wesentlichen Aspekten von konventionellen Workflowsystemen:

- Es gibt eine Dualität zwischen dem Geschäftsprozessmanager und einem Taskmanager. Der Geschäftsprozessmanager verwaltet die Instanzen von Geschäftsprozessen. Für ihn sind alle Ereignisse gleichwertig, egal aus welchem Kanal sie kommen und ob sie von einem EDV-System oder von einem menschlichen Bearbeiter ausgelöst wurden. Der Taskmanager abstrahiert die Aktionen von Menschen in Ereignisse (Events).

- Geschäftsprozesse und die Aktionen und Ereignisse aller DV-Systeme und Bearbeiter sind durch das Kommunikationsmodell gut integriert. Dies ist bei traditionellen Workflow-Managern nicht der Fall. Das Modell von Vitria ist also einfacher, universeller und orthogonaler als die Abstraktionen, die einem normalen Workflow-Manager zugrunde liegen, weil die Abstraktionen normaler Workflow-Manager wenig bis nichts über die Kommunikation mit der Außenwelt aussagen, außer dass Aktivitäten gestartet werden. Asynchrone Kommunikation mit Ereignissen ist die elegantere Abstraktion für das Problem.

Das Prinzip des Taskmanagers ist in Abbildung 92 gezeigt. Der Geschäftsprozess kann eine Aufgabe für einen menschlichen Bearbeiter generieren und legt sich so lange schlafen, bis die Aufgabe erledigt ist. Der Geschäftsprozess wird wieder aufgeweckt, wenn eine Benachrichtigung (Ereignis) vom Taskmanager eintrifft, dass die Aufgabe von einer Person erledigt wurde. Damit ist der Taskmanager eine Abstraktionsschicht oder man kann auch sagen ein Adapter für „menschliches Verhalten".

5.2.2 Vitria und die Referenzarchitektur

Es ist nun nicht überraschend, dass Vitria die Referenzarchitektur abdeckt, die wir in Abschnitt 3.1.8 als Zusammenfassung der Fähigkeiten und Modelle von EAI-Integrationsservern kondensiert haben, die auf dem Markt zu finden sind. Es gibt mehrere Wege, um zu Referenzmodellen für Kategorien von Anwendungen und Lösungen zu kommen. Man kann etwas Neues erfinden. Das ist nicht unser Anspruch. Oder man kann das Existierende zusammenfassen und bewerten. Dies entspricht eher dem Ziel und Anspruch dieses Buches.

Die Architekturen, die einer Branchenreferenz am nächsten kommen, findet man dann im magischen Quadranten (wir erinnern uns an Abbildung 86) tendenziell eher bei den Marktführern.

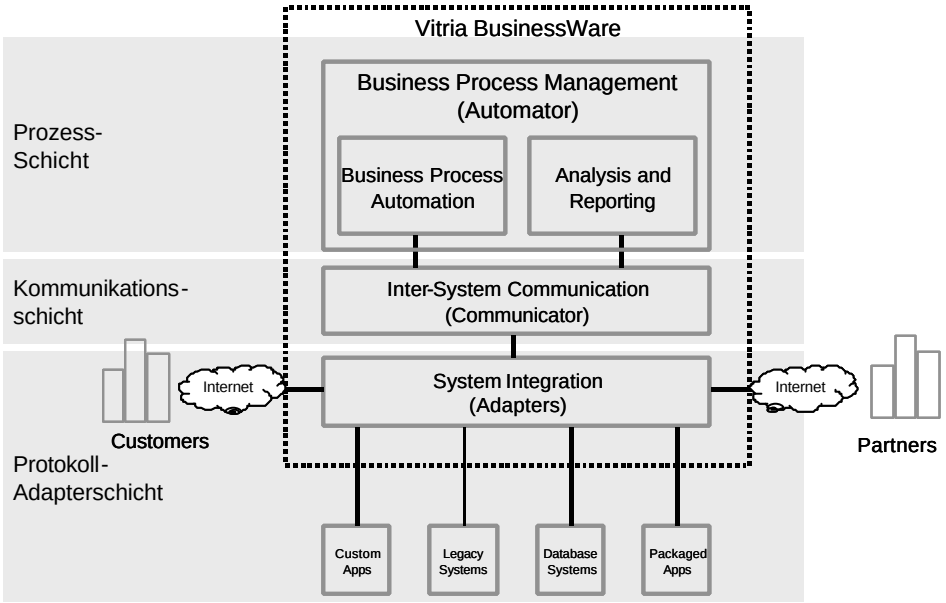


Abbildung 93: Vitria und die Referenzarchitektur

Vom Referenzmodell zur Abdeckung des Referenzmodells kommt man durch den Abgleich der obigen Abbildung 93 mit dem Referenzmodell in Abbildung 23. Es ist unschwer Isomorphie festzustellen, wenn man die Anhänge, die in der Architektur für „Customers" (Kunden) und „Partners" (Geschäftspartner) gemacht wurden, als Adapter sieht.

5.3 Diskussion der Fallstudien

Wenn man beide Fallstudien zusammennimmt, ist die Geschichte, die hinter den Fallstudien steht, nicht untypisch für die Situation vieler Finanzdienstleister. Nicht wenige Unternehmen

haben Mitte der 90er Jahre Workflow eingesetzt. Ebenfalls nicht wenige haben sich die eine oder andere EAI-Komponente gegen Ende der 90er Jahre selbst gebaut oder eine Teillösung zugekauft.

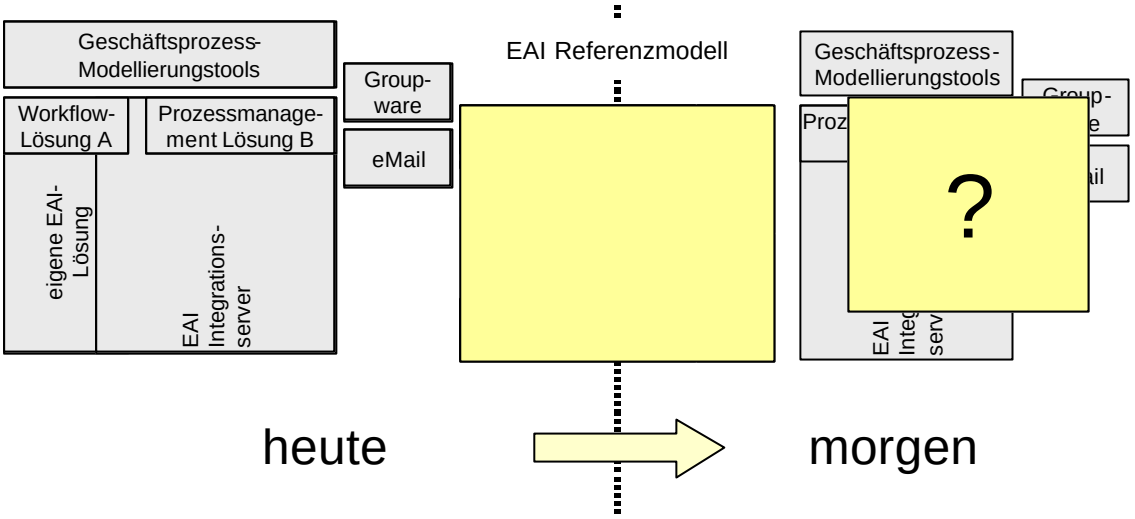


Abbildung 94: Architekturbereinigungen werden bei vielen Finanzdienstleistern nötig werden

5.3.1 Bereinigung von Architekturen

Die linke Seite von Abbildung 94 zeigt die Situation, in der man sich dann heute befinden kann:

- Die Funktionalitäten von Workflow, Groupware, E-Mail überschneiden sich unterschiedlich störend.
- Ebenso überschneiden sich die Funktionalitäten von eventuell schon im Einsatz befindlichen EAI-Integrationsservern mit eigener EAI-Funktionalität.
- E-Mail und Groupware stehen auf der Prozessebene eher „daneben“. Die Funktionalität beißt sich nicht sehr stark mit den Workflow- und Geschäftsprozessmanagement-Lösungen.
- Es gibt ein Geschäftsprozessmodellierungstool wie zum Beispiel ARIS oder Adonis, das mit der Workflow-Lösung über eine Toolkette verbunden ist.

Es ist klar, dass so viel redundante Funktionalität auf die Dauer zu hohe Wartungskosten und zu viel Verwirrung verursacht. Die Verwirrung erhöht wieder die Kosten. Damit werden EAI-Integrationsserver dazu führen, dass viele Unternehmen ihre Softwarearchitektur bereinigen und ändern werden müssen. Dadurch kann es zu folgenden Entwicklungen kommen (siehe rechte Seite der Abbildung 94):

- Die Eigenbaulösungen im EAI-Bereich werden verschwinden und durch zugekaufte Produkte ersetzt werden. Es wird dort eine ähnliche Entwicklung wie bei der Einführung relationaler Datenbanken geben. Produkte werden ältere Lösungen und „Selbstgebautes“ immer mehr verdrängen. Dies ist nicht so schlimm, weil die größten Investitionen nicht in der Kommunikationsschicht stecken (Engines sind billig, Adapter sind teuer, siehe auch

Abbildung 116), sondern in den Adaptern. Die wird man auch in neuen Architekturen weiter verwenden können, zumal wenn sie in Java geschrieben wurden.

- Für Workflow-Manager, die nicht als Prozessmanager nahtlos in die EAI-Infrastruktur integriert sind, wird es sehr eng werden. In Abschnitt 5.2.1.4 konnte man gut das Potenzial erkennen, das man realisieren kann, wenn man Geschäftsprozesse und das EAI-Werkzeug gut miteinander integriert und dafür vor allem ein einziges, einfach zu verstehendes Paradigma hat und die Geschäftsprozesse als getrieben von Ereignissen betrachtet.

- Es gibt auch schon erste Lösungen, wie man seine Werte aus der Geschäftsprozessmodellierung, die in Tools wie ARIS oder Adonis erfolgte, in eine neue Welt retten kann, indem man die Geschäftsprozessdesigntools mit EAI-Lösungen wie Vitria integriert.

- Groupware und E-Mail werden weiter „danebenstehen“, wobei E-Mail über Adapter gut anbindbar ist. Groupware wird sich weiter mit Geschäftsprozess-Ausführungsumgebungen überschneiden. Damit kann man allerdings gut leben.

Ein weiterer, allerdings kleinerer Integrationsschub ist noch absehbar, wenn man den Begriff Enterprise Nervous System (ENS) aus Abschnitt 2.1 wieder betrachtet. Die zwei noch fehlenden Elemente Applikationsserver und Webserver sind allerdings ausreichend leicht „anzudocken“. Ein Anwendungsserver ist im Endeffekt nur ein weiteres System, das Ereignisse (Events) produziert und konsumiert.

5.3.2 Aus welcher Situation komme ich?

An den Fallstudien ist auch deutlich zu sehen, dass der aktuelle Handlungsbedarf immer mit aus der bisherigen Evolution eines Anwendungsportfolios bestimmt wird.

Schöne Lösungen mit orthogonalen Konzepten wird man dann sehr schnell adaptieren, wenn man auf der grünen Wiese beginnen kann. Wer von null mit einem E-Business startet, wird sich nicht sehr schwer damit tun, mit einem gut integrierten EAI-Integrationsserver zu beginnen. Er wird ein Produkt aus dem Quadranten der Marktführer kaufen und auf neuere Technologien setzen.

Wer allerdings einen großen Teil seiner Unternehmensdaten in 25 Jahre alten Hostsystemen aufbewahrt, muss mehr Gedanken an die sinnvolle Integration neuer Welten mit diesen erheblichen Werten verwenden. Wenn dazu noch ca. 7-10 Jahre alte Workflow-Architekturen kommen, muss man noch mehr Umsicht investieren. Hier wird man manchmal versucht sein, Nischenprodukte einzusetzen, um kurzfristig Investitionen zu sichern. Langfristig muss allerdings jeder Prozessintegration betreiben und dann würden die Nachteile von nicht wirklich runder Integration auch bei den Kosten zum Tragen kommen. Lösungen wie EDS und Nischenprodukte werden damit mittelfristig aussterben. Auch hier wird man mit einer wirklich guten, einfach zu verstehenden Architektur besser bedient sein.

Dies ist ein kleiner Vorgriff auf das übernächste Kapitel, in dem wir uns mit Fragen beschäftigen, wie man zu einer sinnvollen EAI-Strategie und Architektur kommt.

6 Microsofts EAI-Strategie

Bisher haben wir uns in diesem Buch, vielleicht ohne dass mancher Leser es bemerkt haben mag, in einer Welt bewegt, die eher von IBM-Hosts und Suns Java dominiert war. Nachdem der Autor dieses Buches in der Finanzindustrie arbeitet, ist das nur natürlich, denn dort sind bei weitem die meisten Server (Bestandssysteme) auf OS/390-Rechnern installiert.

Man kann allerdings kein Buch mehr über EAI schreiben, ohne auch die Technologien von Microsoft zu betrachten. Sie sind quasi die Gegenwelt zu der Welt, mit der wir uns bisher beschäftigt haben. Nachdem sich Microsoft anschickt, nach dem PC-Desktop-Markt das Internet und dort auch den kommerziellen Markt der Unternehmensanwendungen anzugehen, muss man sich ansehen, was Microsoft bieten kann, wenn man ernsthaft über EAI-Technologie reden will. Mit der so genannten BizTalk-Initiative und dem BizTalk-Server stellt Microsoft auch einen EAI-Server zur Verfügung.

Um die Strategie von Microsoft verständlich zu machen, werden wir zunächst in Abschnitt 6.1 einen kurzen Abriss über die .NET-Strategie und Architektur von Microsoft geben. Dies ist sinnvoll, weil die Produkte der .NET-Initiative eng aufeinander abgestimmt sind. Am besten wird man bedient, wenn man nicht eines der Produkte getrennt einkauft, sondern komplett alle, insbesondere wegen der engen Abstimmung und weil sie einander teilweise auch benutzen.

In Abschnitt 6.1.4 gehen wir auch auf Webservices und SOAP ein. Dies ist eine Technologie, um „irgendwo im Web“ Dienste zur Verfügung zu stellen, die einfach aufzurufen sind. Mit .NET-Technologien sind sie sogar besonders einfach aufzurufen.

Zur .NET-Architektur gehören eine Menge fertiger Server, wie zum Beispiel Portalserver, Mailserver, der SQL-Server und mehr. Diese werden wir in Abschnitt 6.1.5 im Überblick vorstellen, bevor wir zu dem Server kommen, der uns im Zusammenhang mit dem Thema dieses Buches besonders interessiert – dem BizTalk-EAI-Server. Dieser wird im Abschnitt 6.2 eingehender betrachtet.

6.1 Eine kurze Einführung in .NET

Zunächst erwartet man sich von einer Einführung in .NET eine Definition. Eine solche Definition zu geben, ist nicht ganz einfach. Wenn man in der Sekundärliteratur nichts Konsistentes findet, macht es immer Sinn, sich die Primärquellen anzusehen. Microsoft stellt .NET auf der Entwickler-Website folgendermaßen vor:

> **Definition .NET:**
>
> Microsoft .NET is an XML Web services platform that will enable developers to create programs that transcend device boundaries and fully harness the connectivity of the Internet.

XML ist schon bekannt, der Begriff Webservices verdient aber ebenfalls eine Betrachtung. „Webservice“ ist auch ein Zauberwort, das volle Seminarräume sichert. Es ist eines der gerade „gehypeten“ silbernen Geschosse, auch Silver Bullets genannt. Dabei kann man die Vorträge relativ schnell beenden, wenn man folgende Definition benutzt:

> **Definition Webservice:**
>
> Ein Webservice ist jede Anwendung oder Anwendungskomponente, die über Standard-Web-Protokolle aufgerufen werden kann.

Damit ist alles, was man über http, https, smtp, SOAP und einige andere Protokolle aufrufen kann, ein Webservice. Man kann also zum Beispiel auch die bekannte Suchmaschine „google“ als Webservice betrachten, wenn man die URL

http://www.google.com/search?q=web+service&hl=de&btnG=Google-Suche&lr=

als den Service sieht und den HTML-Antwortdatenstrom als Ausgabe. Der Service passt allerdings nicht gerade zu .NET, weil weder der Aufruf noch die Antwort mit XML erfolgen.

6.1.1 Die Vision hinter .NET: Die dritte Generation des Internets

Welche Vision steckt also hinter dem Satz „to create programs that transcend device boundaries and fully harness the connectivity of the Internet.“? Dahinter steckt die Vision vom Netz als Computer.

Die erste Generation des Internets hat es im Wesentlichen ermöglicht, statische Dokumente mit einem Browser zu betrachten. Die zweite Generation hat es erlaubt, mit einem Browser auf mehrschichtige Anwendungen (n-Tier-Anwendungen) zuzugreifen. Die Vision von .NET geht in die Richtung, echte verteilte Verarbeitung auf der Basis von Nachrichtenaustausch über das Internet möglich zu machen. Abbildung 95 stellt die erste und die dritte Generation gegenüber.

An der dritten Generation fällt noch auf, dass die Bezieher von Diensten nicht nur Clients mit Browsern sind, sondern beliebige Anwendungen, die sich der Dienste anderer Anwendungen über das Netz bedienen – also echte verteilte Verarbeitung über das Netz. Das mag für Technologen auf den ersten Blick sehr ansprechend klingen. Die Frage ist nur, ob dies auch für EDV-Profis gilt, die einen sicheren Tagesbetrieb gewährleisten müssen. Dazu kann man auch folgende Sichtweise haben:

?? Verteilte Verarbeitung heißt, dass immer irgendetwas nicht verfügbar ist.

?? Wenn alles auf einem Host läuft, weiß ich, wen ich bei einem Fehler anrufen muss. Wenn ich ein heterogenes System habe, schieben sich die Hersteller gegenseitig die Verantwortung zu, und bei einem echt verteilten System weiß ich nicht einmal mehr, wen ich anrufen soll.

Ähnliche Phänomene findet man auch beim Umgang mit anderen mächtigen Werkzeugen, wie objektorientierten Programmiersprachen. Man hat ein mächtiges Werkzeug mit großen Freiheiten, aber man muss es diszipliniert einsetzen und sich selbst Konventionen auferlegen. .NET enthält deshalb Folgendes:

?? Eine Referenzarchitektur, die zeigt, wie man mit der Freiheit und Macht sinnvoll umgehen sollte und welche Beschränkungen man sich auferlegt.

?? Zu .NET gehören außerdem die Werkzeuge, mit denen man solche .NET-Anwendungen und Systeme realisieren kann.

?? Und endlich gehört zu .NET auch eine Menge von fertigen Diensten, die man in Server verpackt direkt einsetzen kann und die einschließlich der Werkzeuge aufeinander abgestimmt sind, weil sie von einem Hersteller stammen.

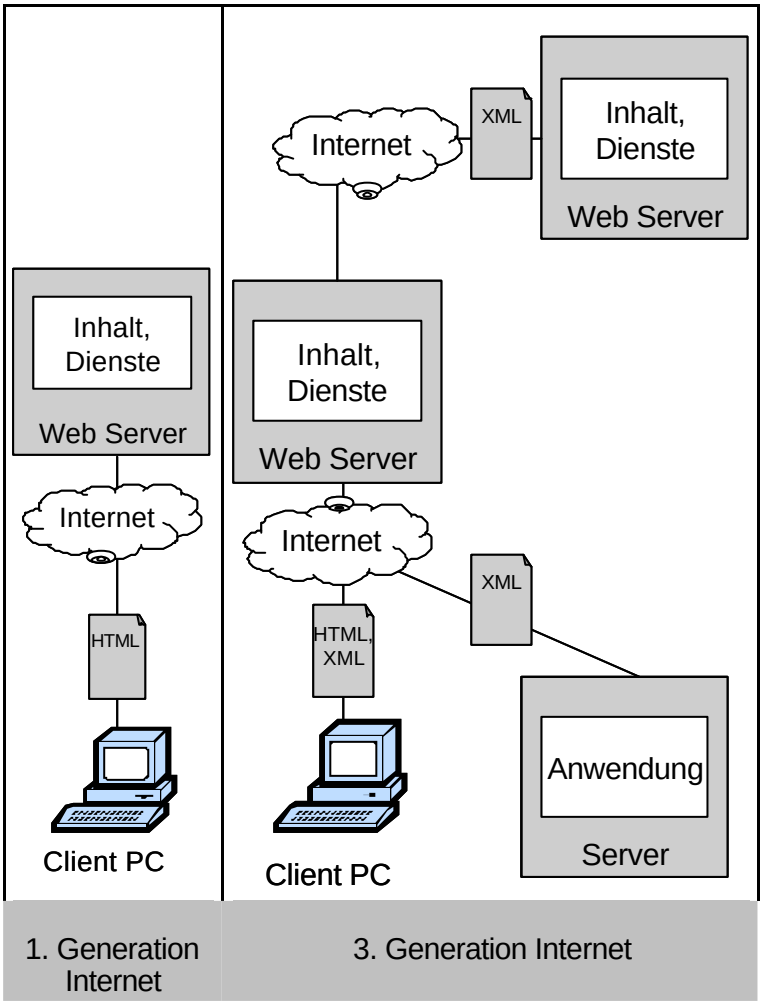


Abbildung 95: Erste versus dritte Generation des Internets

Wir werden in diesem Kapitel also von unten nach oben vorgehen. Es wird erst ein Überblick über die Sprachen und Programmierwerkzeuge gegeben. Dann werden wir den Begriff eines Webservices näher betrachten, um im Folgenden die Server vorzustellen, mit denen .NET fertige Dienste anbietet. Dann sind wir am eigentlichen Ziel und können uns mit BizTalk, dem EAI-Server der .NET-Architektur, befassen.

6.1.2 .NET-Programmiersprachen und Programmierumgebungen

Microsoft hat mit .NET die Idee einer universellen Ausführungsumgebung für viele Programmiersprachen aufgegriffen, die durch Java allerdings aus Gründen der Portabilität schon vorher beschritten worden war. Die entsprechende Laufzeitumgebung (siehe Abbildung 96) heißt bei .NET Common Language Runtime (CLR).

Die CLR bietet Dienste in Form von Basisklassen, Support für die Programmierung mit Threads (leichtgewichtigen Prozessen), COM Marshalling, Typprüfung, Ausnahmebehandlung, Sicherheitsmechanismen, Garbage Collection und andere mehr. Nachdem dieses Buch EAI zum Thema hat, sei für Details auf eines der vielen Werke zu .NET verwiesen, wie zum Beispiel [Vast+2001] oder [Simm+2002].

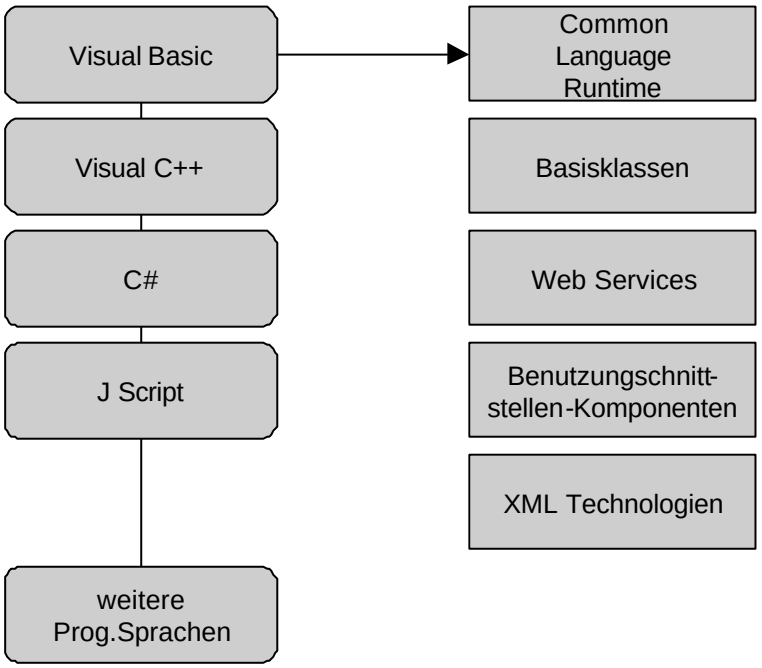


Abbildung 96: .NET-Programmiersprachen und darauf aufbauende Frameworks

Die Programmiersprachen, die Microsoft im Rahmen der .NET-Strategie anbietet, lassen sich sämtlich auf die CLR abbilden. Damit kann zum Beispiel sicher auch eines Tages der Traum vielen Smalltalk-Entwickler in Erfüllung gehen, wieder Smalltalk im Mainstream zu programmieren, weil auch andere Hersteller von Programmiersprachen Sprachen anbieten können, die in den Zwischencode für die CLR übersetzt werden.

Das alleine wäre noch nicht durchschlagend. Microsoft verfügt aber mit dem COM+-Komponentenmodell auch über eine Infrastruktur für verteilte Objekte und hat mit dem Visual Studio eine Entwicklungsumgebung gebaut, mit der sich Web-Anwendungen und normale GUIs gleichermaßen einfach entwickeln lassen.

COM+-Objekte können eingebunden werden und man merkt ihnen nicht mehr an, in welcher Programmiersprache sie ursprünglich erstellt wurden. Vor allem können aber auch Webservices genau so einfach eingebunden werden. Bevor wir uns damit beschäftigen, werfen wir allerdings noch einen kurzen Blick darauf, wie das Referenzmodell für browserfähige Anwendungen in .NET aussieht.

6.1.3 .NET-Referenzarchitektur für Web-Anwendungen

Bevor wir uns ansehen, wie man mit Microsoft .NET Webservices bauen kann, müssen wir zunächst noch ein bisschen tiefer in die .NET-Architektur einsteigen. Das Framework, mit dem in .NET Web-Anwendungen gebaut werden, heißt ASP.NET. Active Server Pages sind vielen Entwicklern einigermaßen bekannt. Nur ist die Assoziation, die das Akronym ASP weckt, nicht unbedingt zielführend, um gut zu verstehen, wie man mit .NET Web-Dialoge und Webservices bauen kann. Um den Unterschied verständlich zu machen, werfen wir daher noch einen Blick auf die „alte“ ASP-Architektur.

6.1.3.1 Active Server Pages vor .NET

Active Server Pages vor .NET waren HTML-Seiten, die mit zusätzlichen, speziellen Tags angereichert waren, die von einem speziellen Seitenprozessor interpretiert wurden und die dazu dienten, die Ausgaben beliebiger (COM-)Objekte in die HTML-Seite zu integrieren, die am Browser angezeigt wurde. Der Code solcher Seiten sah dann zum Beispiel wie folgt aus:

```
<%
'*********************************************************
'*                                                       *
'*     asp Beispiel                                      *
'*                                                       *
'*********************************************************
%>

<%
Dim dStunde
dStunde = Hour(Now)

If dStunde < 12 Then
   Response.Write "Guten Morgen!"
ElseIf dStunde < 17 Then
   Response.Write "Guten Tag!"
Else
   Response.Write "Guten Abend!"
End If
%>
Wir hoffen Ihnen gefaellt diese Seite.<BR>
<BR>
Die aktuelle Zeit ist <%= Time() %> und das Datum ist <%= Date() %>.<BR>
```

Beim Aufruf einer solchen Seite „mypage.asp“ wurde das folgende Szenario durchlaufen:

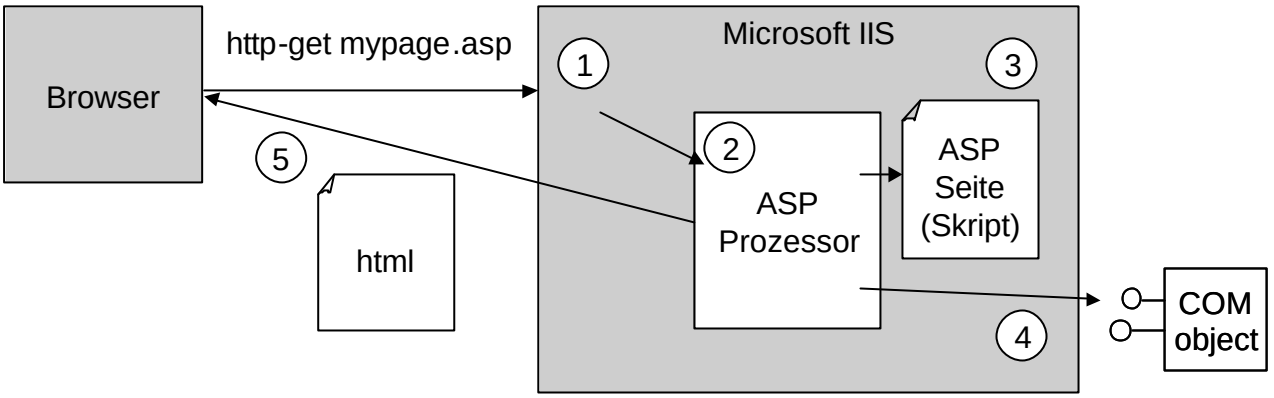


Abbildung 97: Ablauf von ASP alt

Der Browser richtet eine http-get-Anfrage an einen Microsoft Internet Information Server (IIS). Dieser erkennt an der Erweiterung (extension) des Dateinamens, dass eine ASP-Seite aufgerufen wird (1). Der Aufruf wird an einen ASP-Prozessor übergeben (2). Dieser ASP-Prozessor muss zunächst einmal den Code der Seite laden und parsen (3). Dann kann der ASP-Prozessor die Funktionsaufrufe verarbeiten, die in der Seite enthalten sind. Das können auch Aufrufe auf COM-Objekte sein (4). Der ASP-Prozessor ersetzt dann die Einfügestellen mit den Ergebnissen der Ausgaben der in die ASP-Seite eingebetteten Funktionsaufrufe und gibt einen Textstrom zurück, der hoffentlich korrektes HTML ist.

Wenn man .asp durch .jsp ersetzt, IIS durch einen Webserver mit JSP-Engine und die COM-Objekte durch Java-Objekte oder anderes, dann hat man die damals aktuelle Architektur von Sun vor sich. Diese Architektur hatte in beiden Ausprägungen zahlreiche Nachteile:

- Programmlogik und HTML werden vermischt.
- Der ASP-Prozessor parst die Seiten zur Laufzeit und benötigt dafür Prozessorkapazität.
- Dinge wie Sicherheit und Session Handling müssen extra hinzugefügt werden. Bei ASP in der alten Version wurde das mit Cookies gemacht, die nicht jeder Benutzer gerne hatte.
- und weitere ...

Basierend auf diesen Erfahrungen hat Microsoft für .NET eine neue Architektur entwickelt, die zwar jetzt ASP.NET heißt, aber mit der alten Architektur nicht mehr viel gemeinsam hat.

6.1.3.2 ASP.NET

Die Basis für die neue Architektur ist die .NET-Laufzeitarchitektur und eine enge Integration mit Microsofts Entwicklungsumgebung Visual Studio. Wenn man ihn nicht sehen möchte, sieht man so gut wie keinen HTML-Code mehr, sondern arbeitet ähnlich wie mit Visual Basic oder einem anderem Werkzeug für den Bau grafischer Oberflächen.

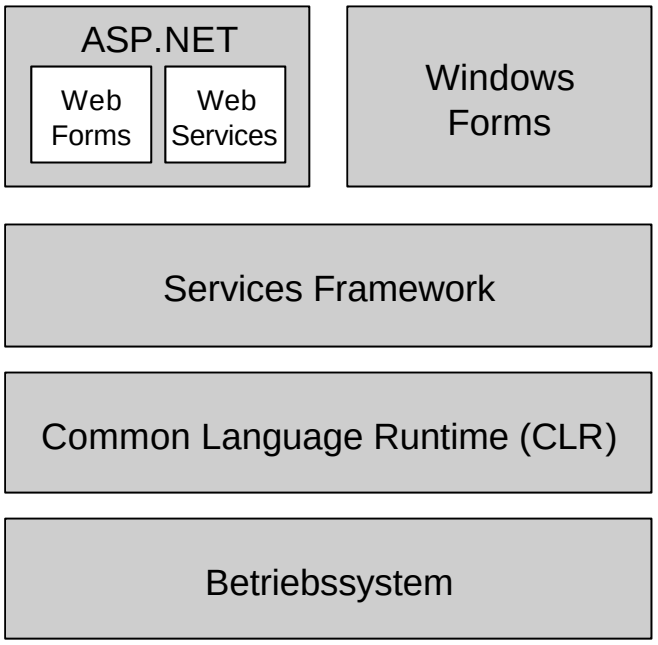


Abbildung 98: ASP.NET-Architektur

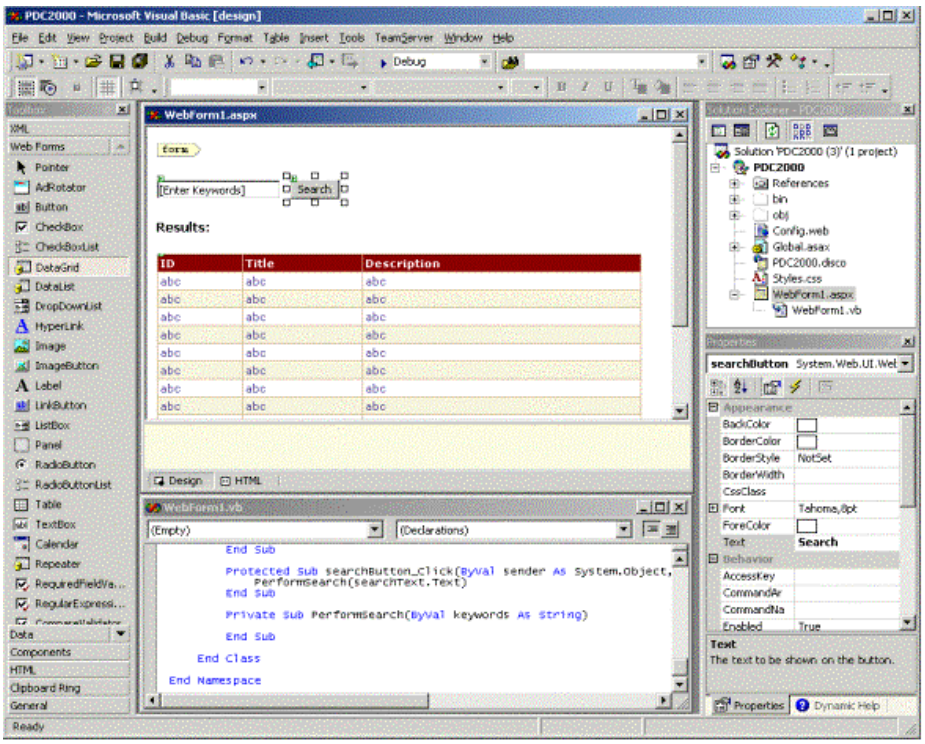


Abbildung 99: Editieren von Web Forms mit Visual Studio (Abbildung Microsoft)

Die Common Language Runtime wurde bereits in Abbildung 96 vorgestellt. Auf dieser Runtime gibt es Service-Frameworks, die eine Menge elementarer Dienste für alle .NET-Anwendungen anbieten. Beispiele dafür sind Datei- und Datenbankzugriffe, Threads und Ähnliches. Abbildung 98 zeigt die Schichten dieser Architektur. Auf die Schicht der Services setzt eine weitere Schicht auf, die es erlaubt, wirkliche Anwendungen zu bauen. Wenn man eine Windows Fat-Client-Anwendung und keine Web-Anwendung bauen möchte, bietet Visual Studio dafür Windows Forms an.

Für Web-Anwendungen kann man ein Programmiermodell verwenden, das sich Web Forms nennt. Auch für Web Forms gibt es einen Interface Builder (siehe Abbildung 99) mit einem ereignisgetriebenen Modell, so wie man es von normalen GUIs gewohnt ist. Damit werden Dinge, wie Sessions, Funktionsaufrufe aus dem HTML-Code und Ähnliches vor dem Benutzer vollkommen versteckt. Auch gibt es keinen Skript-File mehr, der zur Laufzeit jedes Mal geparst und interpretiert wird, sondern Sourcecode, der mit einem Just-in-Time-Compiler übersetzt wird.

6.1.4 Webservices bauen mit SOAP und .NET

In Abbildung 98 haben wir einen Teil der obersten Schicht noch nicht behandelt, nämlich die Webservices, die ebenfalls in ASP.NET integriert sind. Diese können dann über SOAP aufgerufen werden.

Was ist das nun schon wieder? Seife?

Eigentlich müsste man SOAP, das Simple Object Access Protocol, nicht erläutern, wenn man erklären will, wie man mit einer Microsoft .NET-Umgebung Webservices erstellen kann. Mit der Microsoft Entwicklungsumgebung programmiert man „normale“ Objekte und gibt danach die Anweisung, dass diese als Webservice aufbereitet werden.

Der Blick auf SOAP ist also ein Blick auf die Mechanik, wie solche Objekte dann aufgerufen werden können. Sie können auch von Anwendungen aufgerufen werden, die nicht mit .NET programmiert sind.

Kommen wir also zurück zum Begriff SOAP und erinnern wir uns an die Definition eines Webservice in Abschnitt 6.1. Danach ist ein Webservice ein Dienst, der über ein Standardprotokoll, wie zum Beispiel http, aufgerufen werden kann. Am einfachsten versteht man SOAP, wenn man sich einen SOAP-Request einmal ansieht. Der unten stehende stammt aus der Spezifikation [SOAP1.1] und ist in einen http-put-Request eingebettet. Solche SOAP-Nachrichten können allerdings auch mit beliebigen anderen Internetprotokollen transportiert werden, wie zum Beispiel ftp oder smtp.

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:GetLastTradePrice xmlns:m="Some-URI">
           <symbol>DIS</symbol>
       </m:GetLastTradePrice>
```

```
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Der eigentliche SOAP-Aufruf beginnt mit `<SOAP-ENV:Envelope`. Es handelt sich dabei um einen XML-Text. Fachlich soll dieser Beispielaufruf einen Aktienkurs für Disney (Code DIS) beschaffen.

Um zu verstehen, wie ein SOAP-Request aufgebaut sein muss, kann man sich die .xsd-Schemabeschreibung ansehen,

```
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
```

die hinter dem Request steht. Ein Vorteil der XML-Sprachenfamilie ist, dass es dafür mächtige Werkzeuge gibt, so zum Beispiel den XML-Editor XMLSpy, mit dem man sich XML-Schemadefinitionen grafisch ansehen kann.

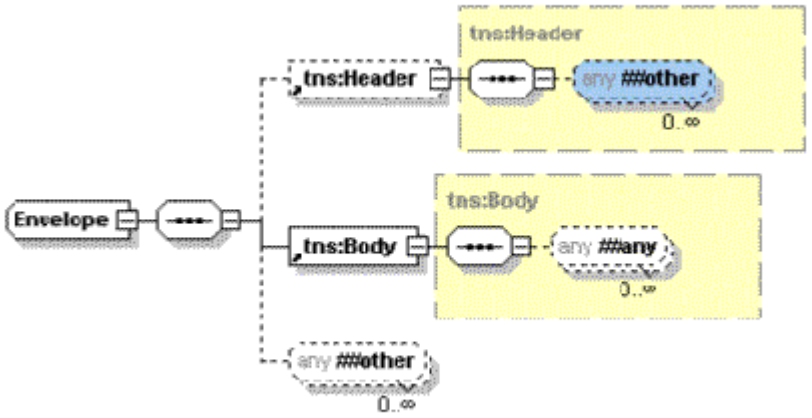


Abbildung 100: Schemadefinition für SOAP-Envelopes

Eine SOAP-Nachricht (Envelope) besteht also aus einem optionalen Nachrichtenkopf, in dem „irgendetwas“ stehen darf, und mindestens einer Nachricht (Body), in der auch „irgendetwas“ stehen darf. Das Irgendetwas ist noch limitiert durch 2 syntaktische Definitionen, von denen eine sichtbar ist.

Erstens kann die Syntax der Parameter durch ein XML-Schema definiert werden, das zu dem Webservice hinterlegt wird:

```
<m:GetLastTradePrice xmlns:m="Some-URI">
```

Und zweitens enthält die SOAP-Schemabeschreibung selbst wieder einen Verweis auf das allgemeine Schema für XML-Schemabeschreibungen:

```
xmlns:xs="http://www.w3.org/2001/XMLSchema/"
```

In diesem kann man nachsehen, wie man in SOAP zum Beispiel komplexe Datentypen beschreiben kann und sollte. Wer sich ein wenig mit Compilerbau auskennt, wird sich in der Abbildung 101 schnell zu Hause fühlen. Wenn Sie damit keine Erfahrung haben, können Sie auch unbeschadet zum nächsten Abschnitt springen und ein spezialisiertes Werk über XML und Schemadefinitionen zu Rate ziehen, wie zum Beispiel [Shep2001].

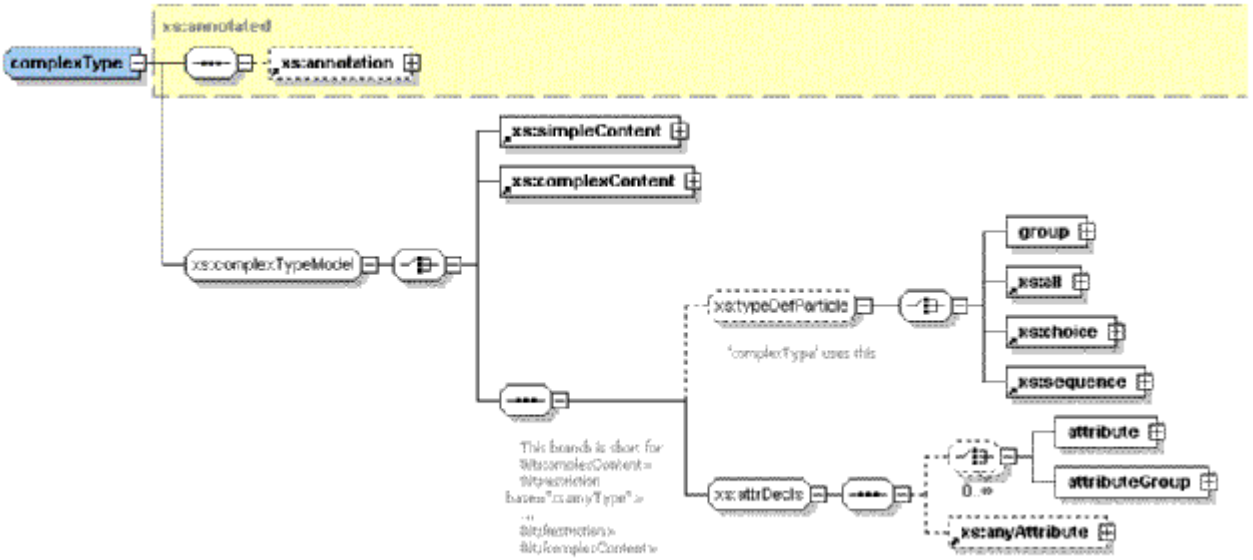


Abbildung 101: Schemadefinition für XML-Schemen (http://www.w3.org/2001/XMLSchema/)

6.1.4.1 Ausführungsmodell

Man kann nun diesen Request über http an einen Webserver schicken, der den geforderten Webservice implementiert, und erhält dann (hoffentlich) die richtige Antwort. Es ist offensichtlich, dass es ziemlich viel Arbeit wäre, einen solchen Webservice ohne die Unterstützung generierender Softwarewerkzeuge zu programmieren.

Man kann sich aber auch vorstellen, dass es kein wirkliches Problem darstellt, aus einer Prozedurdeklaration in einer Programmiersprache wie C#, C++, Java oder Visual Basic einen entsprechenden Service zu generieren.

Microsoft hat genau dies getan und siedelt seine Webservices auf dem IIS (Internet Information Server) an, dem Webserver von .NET. Es lassen sich dann zum Beispiel in Visual Studio normale Objekte programmieren und als Webservices via Generierung zur Verfügung stellen. Die Laufzeitumgebung dafür bietet das ASP.NET-Framework.

6.1.4.2 Benutzung von Webservices

Eine weitere interessante Frage ist noch, was man im Allgemeinen dafür tun muss, um einen Webservice zu benutzen. Man kann natürlich in jeder Programmiersprache die obige XML-Anfrage in einen http-post-Request einbauen und abschicken. Rationell und schnell ist das aber nicht.

Es ist also besser, wenn man sich wieder von Generatoren bei der Arbeit helfen lässt. Damit das sinnvoll funktioniert, sollten Webservices selbstbeschreibend sein. Das heißt, es sollte für den Webservice auf dem Server auch eine Beschreibungsdatei hinterlegt sein, die es einem möglichen Aufrufer des Services erlaubt, sich in seiner Programmierumgebung einen Proxy dafür zu generieren und diesen so zu benutzen, wie ein normales Objekt der Programmiersprache der Entwicklungsumgebung.

Dazu gibt es WSDL, die Webservice Definition Language [WSDL1.1], mit der man als Anbieter eines Webservice diesen beschreiben kann.

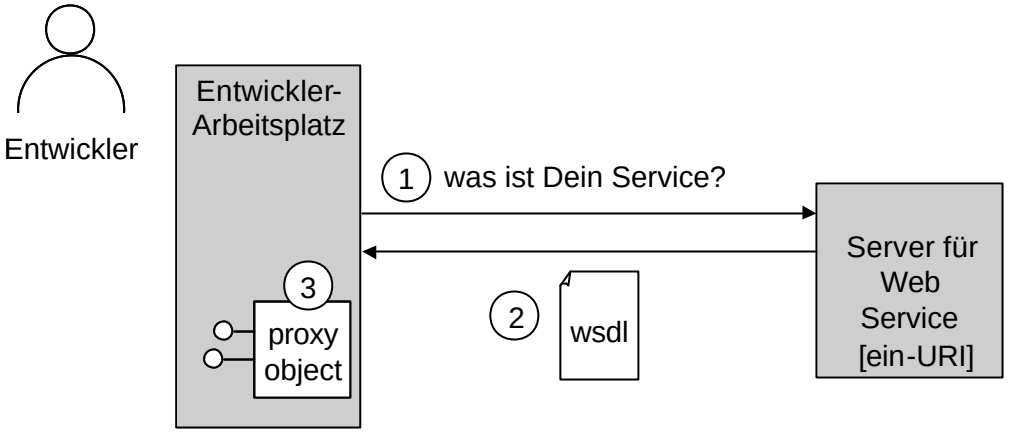


Abbildung 102: Benutzung eines Webservice – Entwicklung

Ein Entwickler wird also über seinen Entwicklerarbeitsplatz einen Server, von dem er weiß, dass dieser einen Webservice anbietet, nach der Beschreibung des Webservice fragen. Abbildung 102, Punkt (1) stellt dies dar. Der Server liefert die Beschreibung in Form einer WSDL-Datei (2). Diese wird vom Entwicklerarbeitsplatz durch Generierung in ein Proxy-Objekt verwandelt und kann in der Programmierumgebung so benutzt werden, als wäre sie ein lokales Objekt.

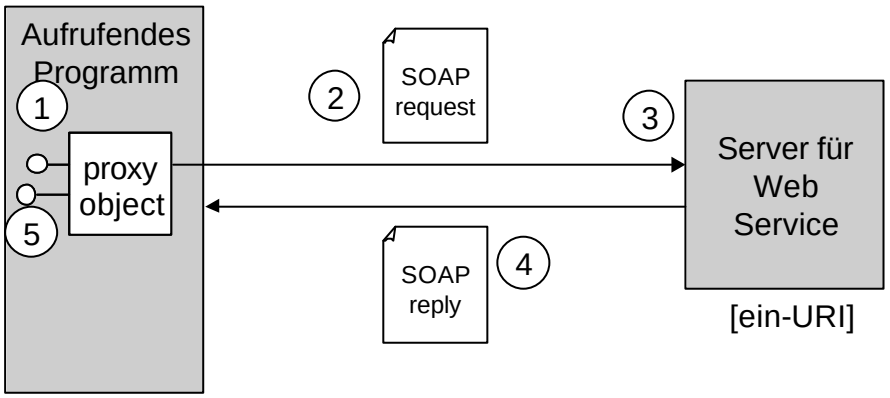


Abbildung 103: Webservice verwenden

Zur Laufzeit benutzt ein Programm den generierten Proxy. Abbildung 103, Punkt (1) stellt dies dar. Der Proxy erzeugt einen SOAP-Request (2) und sendet diesen an den Server. Dieser wird von dem Server verarbeitet (3). Der Server erzeugt eine Antwort (4), die vom Proxy wieder in die internen Formate der Programmiersprache des aufrufenden Programms zurückverwandelt wird.

6.1.5 Server der .NET-Familie und deren Bezug zu EAI

Die .NET-Architektur bietet mehr als einen Server an, um typische Teilaufgaben zu erledigen, die in einer Internet- und E-Business-Architektur anfallen. Wir werden sie hier auflisten und die Aufgaben und Verantwortlichkeiten nennen. Bis auf die Server, die für EAI relevant sind, werden die Server hier nicht näher beschrieben. Bei speziellem Interesse für die .NET-Serverfamilie sei auf Übersichtswerke über alle Server verwiesen (zum Beispiel

[Simm+2002]). Außerdem gibt es für jeden Server eigene Support-Sites bei www.microsoft.com. Die folgenden Server bilden den Kern der .NET-Server.

?? Der **Commerce Server 2000** ist verantwortlich für den Internetauftritt von elektronischen „Geschäften". Dies kann Einzelhandel oder Großhandel sein. Der Commerce Server bietet dazu Fähigkeiten für Produktkataloge, Beobachtung des Surfverhaltens der Benutzer, basierend auf einem Data-Warehouse-Ansatz, und Behandlung von Benutzersitzungen. Benutzer werden durch die Site mit so genannten Pipelines geführt; einer Art von Geschäftsprozessen. Damit ist der Server mit anderen E-Commerce-Servern vergleichbar, die ebenfalls Personalisierung und Analysen des Surfverhaltens bieten.

?? Der **BizTalk-Server** ist der EAI-Server von Microsoft. Er ist eng auf den Commerce Server abgestimmt. Wir werden ihm im Folgenden mehrere Abschnitte widmen, beginnend mit Abschnitt 6.2.

?? Der **Host Integration Server** ist eine Erweiterung zu Microsofts SNA-Server. Er schafft damit die Verbindungen zur IBM Mainframe-Welt und anderen Nicht-Microsoft-Systemen.

?? Der **SQL-Server 2000** wird von den meisten anderen Microsoft-Servern, die Zustände zu verwalten haben, als Datenbankserver genutzt. Auf die Funktionen und die Mächtigkeit eines Datenbankservers gehen wir hier nicht näher ein.

Kenner werden sich jetzt fragen, wo bei all dem der Transaktionsserver MTS (Microsoft Transaction Server) geblieben ist. Der Transaktionsdienst ist als Teil von COM+ in jedem **Windows 2000 Advanced Server** als Teil des Betriebssystems integriert und wird nicht mehr separat gelistet.

Damit ergibt sich das folgende Architekturbild, wenn man mit der .NET-Architektur eine E-Business-Architektur zum Beispiel für einen Einzelhandel im Web aufbauen möchte:

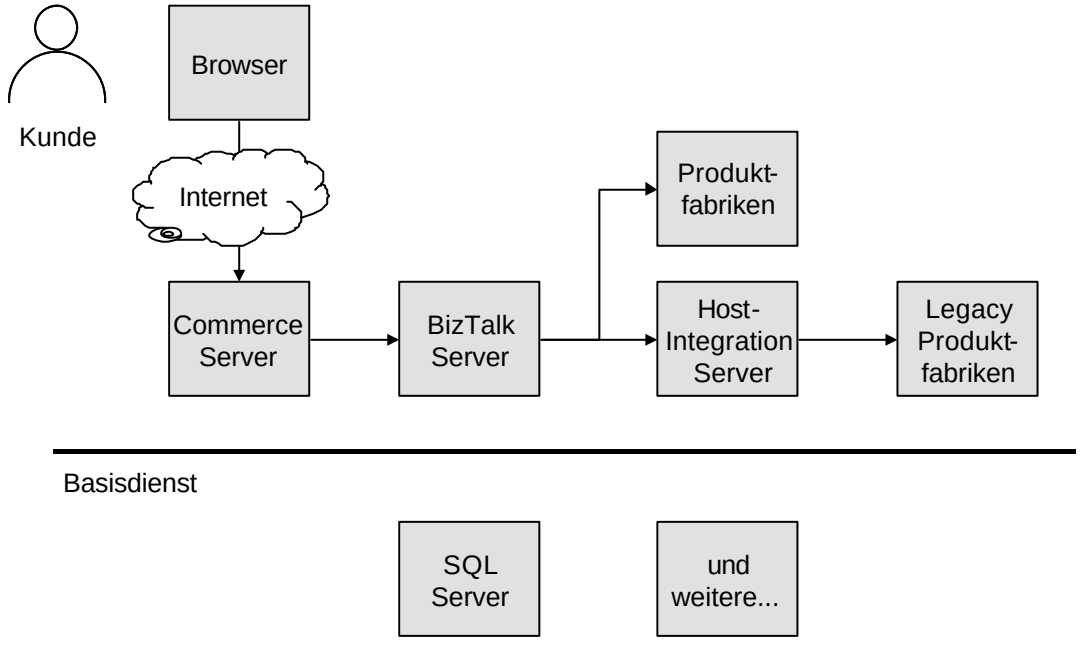


Abbildung 104: Aufbau eines Web-Shops mit der Microsoft-Serverfamilie

Der Kunde benutzt eine Website (siehe Abbildung 104), die mit dem Commerce Server betrieben wird. Wenn der Commerce Server Dienste von Produktfabriken (also Backend- und Bestandssystemen) benötigt, kann er diese auf mehrere Arten anfordern:

- Entweder sind Teile der Dienste über COM+ oder als Webservices erreichbar. Dies ist in der Abbildung nicht gesondert dargestellt.

- Oder die Produktfabrik und andere Backend-Systeme werden über XML-Nachrichten angesprochen. Dann wird der Commerce Server sich des BizTalk-Servers zur Kommunikation bedienen.

- Oder es handelt sich bei der Produktfabrik um ein Hostsystem, das nicht über Schnittstellen verfügt, die schon mit XML so ausgestattet wurden, dass sie für Microsoft-Produkte leicht zugänglich sind. In diesem Fall wird der BizTalk-Server Dienste des Host Integration Servers verwenden, um auf diese Legacy-Produktfabriken zuzugreifen.

Neben den oben gelisteten Kernservern gibt es noch folgende Server, die zusätzliche Dienste erbringen:

- Der **Application Center Server 2000** ist ein Server, der es erlaubt, Servercluster zu bilden und zu verwalten. Zu Details siehe zum Beispiel [Simm+2002].

- Der **Internet Security and Acceleration Server 2000** kann als Proxy-Server, Internet Caching Server und Firewall benutzt werden. Er stellt die Schnittstelle zwischen Ihrem Corporate Network und dem öffentlichen Internet dar.

- Der **Exchange Server 2000** ist der Mail- und Groupware-Server in der .NET Architektur. Der Exchange Server ist lange bekannt, hat sich bewährt und ist in vielen Firmennetzen verbreitet.

- Der **Mobile Information Server** hat die Aufgabe, die Anwendungen des Unternehmensnetzwerkes für mobile Computer (Handhelds, Smartphones, Handys, Laptops) verfügbar zu machen. Die Funktionen sind modular aufgebaut und einsteckbar, so dass es schwer ist, den Server kompakt zu beschreiben. Die Fähigkeiten reichen von E-Mail-Replikation bis zu WAP-Zugang. Darüber hinaus ist der Mobile Information Server mit Basisdiensten wie dem Active Directory und Sicherheitsmechanismen kombinierbar.

- Der **Share Point Portal Server** ist der Portalserver der .NET-Familie.

Einen direkten Bezug zu EAI hat damit die Kombination aus dem BizTalk-Server und dem Host Integration Server. Der Host Integration Server bewegt sich im EAI-Referenzmodell auf der Ebene von Adaptern.

6.2 Der BizTalk-Server

Hinter dem Wort BizTalk verbirgt sich, wie so oft, mehr als eine Bedeutung:

- Zum einen gibt es eine Organisation dieses Namens, die stark von Microsoft unterstützt wird und die sich der Anwendung von XML im E-Commerce verschrieben hat. Auf der Website www.biztalk.org finden Sie unter anderem eine Suchmaschine für XML-Schemata zum Auffinden von Nachrichtenschemata aus den verschiedensten Branchen.

?? Zum anderen gibt es Konventionen für XML-Messages, die ebenfalls unter dem Begriff BizTalk geführt werden. Es handelt sich dabei vor allem um die Konvention, dass bei BizTalk ein Dokument (Document) in einem Umschlag (Envelope) verpackt ist, der es ermöglicht, die Dokumente zu routen (siehe Abbildung 105). BizTalk-Nachrichten sind spezielle Instanzen von SOAP-Nachrichten, die wir in Abschnitt 6.1.4 kennen gelernt haben.

?? Und endlich verbirgt sich dahinter der so genannte BizTalk-Server, bei dem es sich um einen voll funktionalen EAI-Integrationsserver handelt, wie wir an dem Abgleich dieses Servers mit dem Referenzmodell noch sehen werden.

Wie der Begriff BizTalk nahe legt, ist die primäre Stoßrichtung des BizTalk-Servers die Automatisierung des Nachrichtenverkehrs zwischen Unternehmen. Das wesentliche Einsatzgebiet ist B2B-Kommunikation von Unternehmen, die über XML kommunizieren. Primär unterstützte Formate für den Nachrichtenaustausch sind XML, EDI und Textdateien.

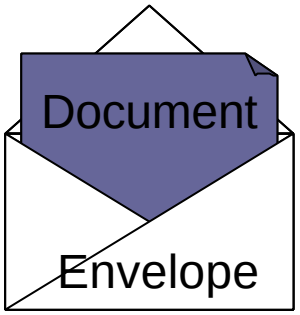


Abbildung 105: BizTalk-Konvention für XML-Nachrichten – oberste Ebene

Die EAI-Fähigkeiten ergeben sich daraus, dass sich mit einem solchen B2B-Server auch schon die meisten Anforderungen an einen typischen EAI-Server abdecken lassen. Der Vergleich von Abbildung 1 und Abbildung 2 legt das auch nahe. B2B-Integration kann man demnach auch als EAI über Unternehmensgrenzen sehen oder umgekehrt kann man EAI als unternehmensinterne B2B-Integration betrachten. Die Abbildungen sind quasi isomorph. Die EAI-Fähigkeiten ergeben sich weiter daraus, dass es mit dem Host Integration Server möglich ist, auch auf die SNA-Welt zuzugreifen. Dazu kommt noch, dass es einen so genannten Orchestration Designer und eine dazu passende Engine gibt, die es erlauben, Geschäftsprozesse zu definieren, die durch Nachrichten getrieben werden und selbst Nachrichten abschicken können. Sie können damit jeden beliebigen Dienst der .NET-Architektur aufrufen.

6.2.1 Begriffswelt von BizTalk

Wenn man sich das Begriffssystem von BizTalk in der Dokumentation (siehe Abbildung 106) ansieht, kann man die Abstammung aus der Kommunikation zwischen Unternehmen deutlich nachvollziehen.

Abbildung 106: Begriffe zur Nachrichtenübertragung mit BizTalk (Quelle [Leyb+2001])

Einen etwas anderen Überblick über die Begriffe gibt Abbildung 107. Mittels BizTalk kommunizieren Organisationen. Diese schicken ihre Nachrichten an einen Channel. Die Nachrichten können aber auch vom BizTalk-Server aktiv abgefragt werden. Der BizTalk-Server ist in der Lage, Nachrichten zu analysieren, zu transformieren und wieder zu verteilen.

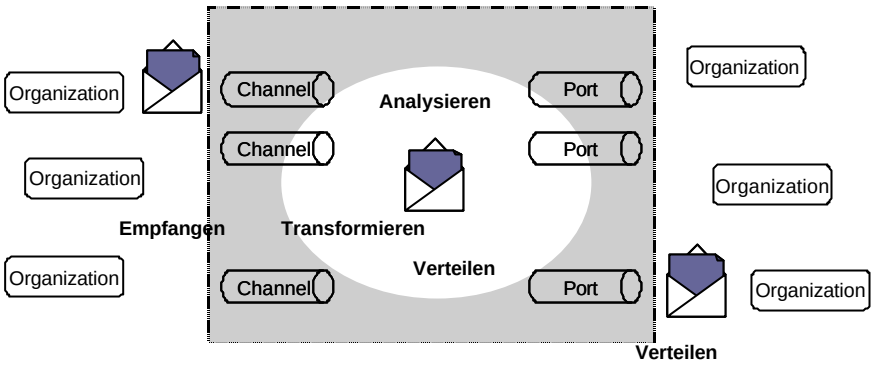


Abbildung 107: Weitere Begriffe zur Kommunikation mit BizTalk

Die Nachrichtensenken werden in der BizTalk-Terminologie Ports genannt. Eine Nachricht kann selbstverständlich an mehr als einen Port gesendet werden. Hinter einem Port steht wieder eine Organisation, die die Nachricht empfängt.

In diesem Bild ist noch nichts von Geschäftsprozessen zu sehen. Das war auch schon bei Vitria so. Wie die Geschäftsprozesse mit dem Nachrichtentransportmodell verbunden sind, könnte jeweils einfacher dargestellt sein. Im Fall des BizTalk-Servers können, wie schon bei Vitria beschrieben, Geschäftsprozesse als Sender und Empfänger von Nachrichten auftreten. Folgende Abbildung zeigt dies mit den Mitteln des BizTalk-Servers. Das Bild stammt aus dem so genannten Orchestration Designer, dem Designwerkzeug für Geschäftprozesse im Kontext von BizTalk.

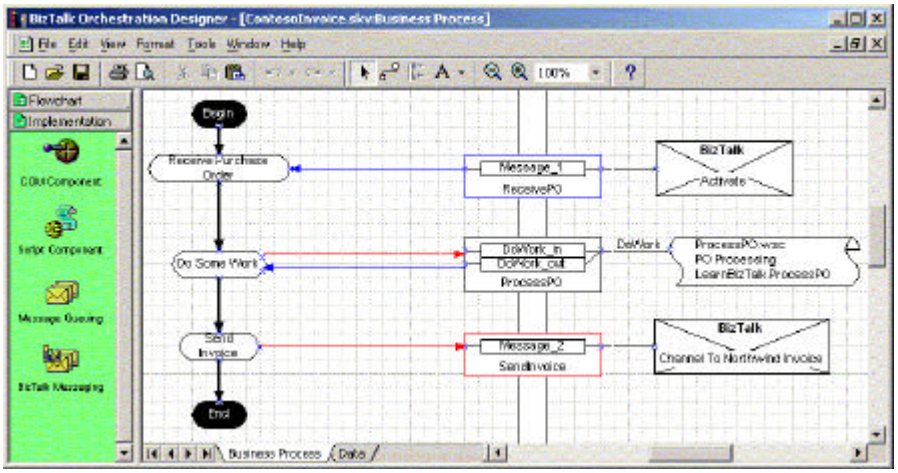


Abbildung 108: BizTalk Orchestration Designer (Quelle [Leyb+2001])

11 Literatur

[Arne1999] Chuck Arney: Web Enabling Native VSE Applications Using Web/VSE-Host, VSE/ESA Software Newsletter, Issue #18**,** *VSE/ESA Software Newsletter* (First/Second Quarter, 1999).

[Bass+1998] Len Bass, Paul Clements, Rick Kazman: Software Architecture in Practice. Addison-Wesley 1998.

[BizNet] Microsoft Corporation, biztalk.NET Website: http://www.biztalk.net/, Ist die Homepage der BizTalk-Initiative.

[BizTalk] Microsoft Corporation, BizTalk Website: http://www.microsoft.com/biztalk. Enthält Informationen zum BizTalk-Server.

[Brod+1995] Michael L. Brodie, Michael Stonebreaker: Migrating Legacy Systems. Morgan Kaufmann Publishers 1995.

[Dada1999] Peter Dadam: Verteilte Datenbanken und Client/Server-Systeme. Grundlagen, Konzepte und Realisierungsformen. Springer-Verlag 1999.

[Date1994] C. J. Date: An Introduction to Database Systems (6th edition). Addison-Wesley 1994.

[DBFo1995] „Vom Eigenbau zum Produkt: Normiertes Verbindungsstück“. In: Datenbank Fokus, Heft 2/95, S. 30 – 34.

[Den1991] Ernst Denert: Software-Engineering. Springer 1991.

[Gam+1994] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns; Elements of Reusable Object-Oriented Software. Addison-Wesley 1994.

[Gray1981] Jim Gray: The Transaction Concept: Virtues and Limitations. Proc. 7th International Conference on VLDB, Cannes, 1981, pp. 144-154.

[Gray+1993] Jim Gray, Andreas Reuter: Transaction Processing: Concepts and Techniques. Morgan Kaufmann Publishers 1993.

[Grif1998] Frank Griffel: Componentware. dpunkt.verlag 1998.

[Härd+1983] Theo Härder, Andreas Reuter: Principles of Transaction-Oriented Database Recovery. ACM Computing Surveys, Vol. 15, No.4, Dec. 1983, pp. 287-317.

[Hous2001] Peter Houston: Selecting Between Synchronous and Asynchronous Alternatives. White Paper, EBizQ Forum (http://e-serv.ebizq.net), 2001.

[IBM1994] DataJoiner – A Multidatabase Server. IBM White Paper 10/94, Reihe „Data Management Solutions“.

[IBM1995] Distributed Relational Database Architecture, Every Manager's Guide, IBM Document # GC26-3195-01, Second Edition, 1995.

[IBM2001] IBM Corp, IBM DCE for AIX, das Handbuch ist über diese Schlüsselbegriffe mit Suchmaschinen an vielen Stellen im Web online verfügbar und einfach mit einer Suchmaschine zu finden. Der für dieses Buch verwendete Hyperlink ist http://www.hpc2n.umu.se/doc/aix/dce/A3U2S/A3U2SM02.HTM.

[Juric2002] Matjaz B. Juric: Professional J2EE EAI. Wrox Press 2002.

[Kell1999] Wolfgang Keller: The Pitfalls of Meta-Systems and Business Rules. White Paper: Background Material for Tutorial "How to Build Flexible Insurance Systems" at STJA99. http://www.objectarchitects.de/ObjectArchitects/papers/WhitePapers/index.htm.

[Kell2001] Wolfgang Keller: A Few Patterns for Managing Large Application Portfolios. Proceedings EuroPLoP 2001 oder auch unter http://www.objectarchitects.de/ObjectArchitects/papers/Published/index.htm .

[Kimb+1998] David Kimball, Laura Reeves, Margy Ross, Warren Thornthwaite: The Data Warehouse Lifecycle Toolkit. Wiley 1998.

[Leyb+2001] Igor Leybovich, Scott Woodgate: Learning BizTalk-Server 2000. Microsoft Corp. 2001, WebTutorial.

[Lint2001] David Linthicum: B2B Application Integration. Addison-Wesley 2001.

[Loos+1997] Chris Loosley, Frank Douglas: High-Performance Client/Server. A Guide to Building and Managing Robust Distributed Systems. Wiley 1997.

[Mano+2001] Dragos A. Manolescu, Adrian E. Kunzle: Several Patterns for eBusiness Applications. Proceedings PloP 2001.

[Musil2000] Sabine Musil (Hrsg.): no-insure.com, Warum Versicherungen im Internet doch eine Chance haben. Karlsruhe, VVW, 2000. XIV, 119 S. (Leipziger Schriften zur Versicherungswissenschaft; 2) ISBN 3-88487-855-7.

[Nuss2000] Richard Nußdorfer: Das EAI-Buch. Erschienen im Selbstverlag, erhältlich beim Autor als elektronisches Dokument, CSA Consulting 2000.

[OMG2001] Jon Siegel: What's Coming in CORBA 3, OMG 2001, siehe auch http://www.omg.org/technology/corba/corba3releaseinfo.htm.

[Open1999] Open Group Technical Standard, DRDA, Version 2, Volume 1: Distributed Relational Database Architecture (DRDA) (C911).

[Orfa+1996] Robert Orfali, Dan Harkey, Jeri Edwards: The Essential Distributed Objects Survival Guide. Wiley 1996.

[Pude2001] Arno Puder: CORBA Feature Implementation Matrices, http://www.ap-c.org/corba/matrix/, based on profiles submitted by users or companies, 2001.

[Ruh+2001] William A. Ruh, Francis X. Maginnis, William J. Brown: Enterprise Application Integration. Wiley 2001.

[Same1997] Johannes Sametinger: Software Engineering with Reusable Components. Springer-Verlag 1997.

[Shar+2001] Rahul Sharma, Beth Stearns, Tony Ng: J2EE Connector Architecture and Enterprise Application Integration. Addison-Wesley 2001.

[Shep2001] Devan Shepherd: XML in 21 Tagen. Verlag Markt und Technik 2001.

[Simm+2002] Curt Simmons, Ash Rofail: The Microsoft .NET Platform and Technologies. Prentice Hall 2002.

[SOAP1.1] Box, Ehnebuske, Kakivaya, Layman, Medelson, Nielsen, Thatte, Winer: Simple Object Access Protocol (SOAP) 1.1, W3C, 8. Mai 2000, http://www.w3.org/tr/soap/.

[Star2002] Gernot Starke: Effektive Software-Architekturen. Ein praktischer Leitfaden. Hanser Verlag 2002.

[Szyp1997] Clemens Szyperski: Component Software. Addison-Wesley 1997.

[VAA1999a] Autorenteam Datenmanager, VAA – die Anwendungsarchitektur der Versicherungswirtschaft, Datenmanager / Anhang. GDV 1999, zu beziehen über http://www.gdv-online.de/vaa/.

[VAA1999b] Autorenteam VAA-Arbeitskreis: Die Technische Beschreibung der pVAA. GDV 1999, zu beziehen über http://www.gdv-online.de/vaa/ .

[Vast+2001] Vasters, Oellers, Javidi, Jung, Freiberger, DePetrillo: .NET Crashkurs. Microsoft Press 2001.

[vHal2001] Barbara von Halle: Business Rules Applied. Wiley 2001

[Vitr2000] Vitria Technology Inc: Business Ware Foundations, Version 3.1., Nov 29, 2000.

[Voel+2002] Voelter, Schmid, Wolff: Server Component Patterns – Component Infrastructures Illustrated with EJB. Wiley 2002.

[Wake2000] Philip Wakelin: Securing Web Access to CICS. IBM Redbook, 2000, kostenlos beziehbar über www.redbooks.ibm.com.

[Wein1985] Gerald M. Weinberg: The Secrets of Consulting. Dorset House Publishing 1985.

[Wiek1999] John-Harry Wieken: Der Weg zum Data Warehouse. Addison-Wesley 1999.

[WSDL1.1] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana: Web Services Description Language (WSDL) 1.1, W3C, 15. März 2001, (http://www.w3.org/TR/wsdl)